On Channel and Transport Layer aware Scheduling and Congestion Control in Wireless Networks

A thesis submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

by

Hemant Kumar Rath
(Roll No. 04407001)

Advisor: Prof. Abhay Karandikar

Department of Electrical Engineering
Indian Institute of Technology Bombay
Powai, Mumbai, 400076.
July 2009

Babu, *Bina*, *Sibu* and *Bisu*
the torchbearers of the next generation

Indian Institute of Technology Bombay

Certificate of Course Work

This is to certify that **Hemant Kumar Rath** (Roll No. 04407001) was admitted to the candidacy of Ph.D. degree in July 2004, after successfully completing all the courses required for the Ph.D. programme. The details of the course work done are given below.

S.No	Course Code	Course Name	Credits
1	EE 740	Advanced Data Network	6
2	IT 605	Computer Networks	6
3	EE 612	Telematics	6
4	IT 610	Quality of Service in Networks	6
5	EES 801	Seminar	4
		Total Credits	**28**

IIT Bombay
Date:

Dy. Registrar (Academic)

Abstract

In this thesis, we consider resource allocation schemes for *infrastructure-based multipoint-to-point wireless networks like IEEE 802.16 networks* and *infrastructure-less ad-hoc networks*. In the multipoint-to-point networks, we propose channel and Transport layer aware uplink scheduling schemes for both *real-time* and *best effort* services, whereas in ad-hoc networks, we propose channel aware congestion control schemes for best effort services.

We begin with the performance evaluation of IEEE 802.16 networks by conducting experiments in the current deployed IEEE 802.16 networks of a leading telecom operator in India. We observe that (i) the throughput of Transmission Control Protocol (TCP) based application is poor as compared to that of User Datagram Protocol (UDP) based application and (ii) TCP-based application suffers in the presence of simultaneous UDP-based application. The Weight-based (WB) scheduler employed in this service provider network does not provide any delay and throughput guarantee to these applications. This motivates us to further investigate scheduling schemes specific to real-time and best effort services (TCP-based applications).

For real-time services, we propose a variant of deficit round robin scheduler, which attempts to schedule uplink flows based on deadlines of their packets and at the same time attempts to exploit the channel condition opportunistically. We call this as "Opportunistic Deficit Round Robin (O-DRR)" scheduler. It is a polling based and low complexity scheduling scheme which operates only on flow level information received from the users. Our key contribution is to design scheduling mechanism that decides how many slots should be assigned to each user based on its requirement and channel condition. Since O-DRR employs no admission control and does not maintain packet level information, strict delay guarantees cannot be provided to each user. Rather, O-DRR reduces deadline violations compared to the Round Robin (RR) scheduler by taking the deadlines of Head of the Line (HoL) packets into account. For fairness, the O-DRR scheduler employs the concept of deficit counter similar to that in Deficit Round Robin (DRR) scheme. We demonstrate that O-DRR achieves lesser packet drops and higher

Jain's Fairness Index than RR at different system loads, fading and polling intervals.

We then consider design of uplink scheduling schemes for TCP-based applications in IEEE 802.16 networks that provide greater throughput and fairness. We propose two variants of RR scheduler with request-grant mechanism: "TCP Window-aware Uplink Scheduling (TWUS)" and "Deadline-based TCP Window-aware Uplink Scheduling (DTWUS)". We consider congestion window $cwnd$, Round Trip Time (RTT) and TCP timeout information of the TCP-based applications while scheduling. To avoid unfairness due to scheduling based only on these information and opportunistic condition, we employ deficit counters similar to that in DRR scheme. Since Adaptive Modulation and Coding (AMC) can be used to achieve high spectral efficiency on fading channels, we exploit AMC for TCP-aware scheduling by extending TWUS and DTWUS with adaptive modulation. Using AMC, the base station can choose a suitable modulation scheme depending on the Signal to Noise Ratio (SNR) of the users so that overall system capacity is increased. We demonstrate that TCP-aware schedulers achieve higher throughput and higher Jain's Fairness Index over RR and WB schedulers. We also discuss the properties of TCP window-aware schedulers and provide an analysis for TCP throughput in IEEE 802.16 and validate this through simulations. For fairness, we observe that the difference of data transmitted by two back logged flows in any interval is bounded by a small constant.

Finally, for ad-hoc networks, we propose a channel aware congestion control scheme, which employs both congestion and energy cost associated with the links of a network. In this scheme, transmission power in a link is determined based on the interference, congestion and available power for transmission of the wireless nodes and the rate of transmission is determined based on congestion in the network. By controlling transmission power along with congestion control, we demonstrate that congestion in the network as well as average transmission power can be minimized effectively. We investigate the convergence of the proposed scheme analytically and through simulations. We then propose an implementation method using modified Explicit Congestion Notification (ECN) approach involving both ECN-Capable Transport (ECT) code points 0 and 1. It uses 2-bit ECN-Echo (ECE) flag as a function of both average queue size and transmission power instead of the usual 1-bit ECE flag.

To summarize, we propose channel and Transport layer aware scheduling schemes for multipoint-to-point networks and channel aware congestion control schemes for ad-hoc networks.

Contents

Acknowledgments

The present thesis would be incomplete without mentioning my heartfelt obligations towards all those people who have contributed in the process of my doctoral research. I am extremely grateful towards **Prof. Abhay Karandikar** for his unceasing support and guidance. Without his insights, constant help and encouragement, this thesis would not have been possible. Prof. Karandikar has a deep influence on my thinking and work through his core values of life, his expertise in technical and written communications and his positive approach towards academic issues. He has always encouraged me to question and look at my own writing critically, which is one of the most essential aspect of a researcher's work. Apart from handling my journey from the thesis conceptualization up to the writing process, he has constantly supported me during all my academic and funding needs. Starting from international and national conferences to fellowships, Prof. Karandikar made sure that I never face any infrastructural inhibition in my academic venture. His mentorship has been a fulfilling experience for me during all these years.

I am highly grateful towards my Research Progress Committee Members for their greatly valuable comments. Both Prof. Anirudha Sahoo and Prof. Prasanna Chaporkar have enriched my research with their critical comments during my Progress seminars and also during individual sessions. I am especially thankful to Prof. Chaporkar for his help during the thesis writing stage. His suggestion to "build a story" before writing helped me to structure the thesis and sharpen its edges.

I am immensely thankful to Prof. Vishal Sharma for his strong support and confidence in my research abilities. Without his encouraging presence and continuous interactions, some papers which are a part of this thesis would not have been possible. I humbly thank Prof. R. N. Biswas for his trust in my academic capabilities and his confidence in me as an individual for more than a decade now. His galvanizing presence in my life has added to my development as a researcher and as a human being.

I am greatly obliged towards Prof. J. Vasi, Prof. V. M. Gadre, Prof. S. A. Soman, Prof. S. Chaudhuri, Prof. A. Q. Contractor, Prof. Amarnath, and Prof. Supratim Biswas, for their incessant concern and warmth. Each of them has inspired my thoughts and my work in their own unique ways and unique set of principles and values. I shall always cherish the tea-time sessions with Prof. Soman as one of my invaluable memories of life.

Infonet Lab has been my second home in IIT Bombay during my M.Tech and PhD years and I feel proud to have seen it grow through all these years. I thank all the Infonet lab members for being with me during the best and worst phases of my research. They are not only my colleagues but also my companions encouraging me throughout. I must mention Abhijeet Bhorkar for providing valuable insight and for being a co-author in one of the papers which is also a part of this thesis. I wish to specially thank Dr. Ashutosh, Dr. Nitin, Punit, R. A. Patil and Brijesh for their critical comments and constant support during the thesis writing phase. I have really gained a lot from the energetic discussions that I have had with all of them during my entire stay at IIT Bombay, as well as during difficult moments of thesis writing, paper drafting, and technical blockages. They made the difficult journey full of fun and energetic. I must thank Prateek for his critical comments and Balaji and Atit Parekh for their critical comments and editorial discussions and fun in the lab. I thank Hrishikesh, Somya and Sonal for making the editorial process such a smooth job for me. Without their help, the intricate technical and editorial jobs would have been extremely tough for me. I am very grateful to Sangeeta and Laxman for their valuable help during and after lab hours.

I thank Arnapurna, a.k.a. Eku a.k.a. Batak for listening to all cribs and learning even TCP/IP literature. She has been my worst critic in the institute, pointing out my grammatical and editorial mistakes both in written and spoken English. But she also has helped me keep cool during the crucial moments of my doctoral work. I am thankful towards my personal physicians Dr. Samir and Dr. Mihir, for being just a phone call away. As friends for the last twenty years, we have shared a lot and they have taken every possible care of my health whenever needed.

I am extremely thankful to my friends and colleagues, Sandeep (Swamiji), Dr. Amey, Swanand, Priyadarshanam (PD), Edmund, Shanmuga, Ashish for their positive influence and fun-filled support. They have always been there with me whenever and whatever time I needed them. I sincerely thank Shivkumar and Mukundan, for facilitating and immensely helping me during the frustrating process of job search. I am indebted towards the entire Team RSF 2007-

08 for their belief and trust in me as their coordinator. We had an amazing time together while organizing events in the institute.

I am deeply obliged towards Atul Amdekar, Naveen Pendyala and Binod Maji for their help during the experimentation in-spite of their busy schedules. I am deeply thankful towards IIT Bombay, Philips India and TICET, for providing me with TA-ship, fellowship and conference funding, without which my research would be incomplete. I am grateful towards the entire EE Office staff, Computer Center, Library, Academic Office, Deans' Offices, Accounts and Cash Sections, for their help and assistance whenever sought for. I am thankful towards Mr. Jore and Mr. Anand for making my stay at Hostel-12, such an enjoyable experience. I also wish to thank Hostel-12 Printout center and Gulmohar for their services.

It is difficult to express my gratitude towards my parents and my family for their unfettered belief on my dreams. I thank my parents for being with me not only during the most difficult phases of my journey but also being there at every juncture of my life. I am grateful towards my elder brothers and my sister for letting me pursue my dreams. A special thanks to both my sister-in-laws for enriching my family with their love and concern and also for the special dishes while at home. My days at Khopoli with my brother and sister-in-law and my play-hours with little Bidhu Bhushan (Shibu), have always eased the loads of a hectic lifestyle. My sister and brother-in-law and their two little children have significantly contributed with their warmth and concern in keeping the "humane" aspect of my personality alive. I would never have thought of my present position in life without my family's support and kindness.

While acknowledging all those who have contributed in my years of research, I miss those who have been left half-way in my journey. I miss both my paternal and maternal grandparents who would have been happiest to have seen me complete the thesis today and also miss some of my friends who have been untimely snatched away from me.

Finally, my deepest gratitude towards the Almighty for making this thesis possible.

Hemant Kumar Rath

July 2009

List of Acronyms

3GPP2	Third Generation Partnership Project 2
3GPP-LTE	Third Generation Partnership Project Long Term Evolution
AIMD	Additive Increase and Multiplicative Decrease
a.k.a	also known as
AMC	Adaptive Modulation and Coding
AQM	Active Queue Management
ARQ	Automatic Repeat reQuest
ARED	Adaptive Random Early Drop
ATCP	Ad-hoc TCP
ATM	Asynchronous Transfer Mode
AWGN	Additive White Gaussian Noise
BE	Best Effort
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
BS	Base Station
BWA	Broadband Wireless Access
CBR	Constant Bit Rate
CC-CC	Channel aware Congestion control with Congestion Cost
CC-NC	Channel aware Congestion control with Network Cost
CDMA	Code Division Multiple Access
CE	Congestion Experienced
CID	Connection IDentifier
CINR	Carrier to Interference-plus-Noise Ratio
CPS	Common Part Sublayer

CS	Congestion Sublayer
CSD-RR	Channel State Dependent Round Robin
CWR	Congestion Window Reduced
DAMA	Demand Assignment Multiple Access
DC	Deficit Counter
DCD	Downlink Channel Descriptor
DFPQ	Deficit Fair Priority Queue
DL_{MAP}	Downlink Map
DRR	Deficit Round Robin
DTPQ	Delay Threshold Priority Queue
DTWUS	Deadline-based TCP Window-aware Uplink Scheduling
ECE	ECN-Echo
ECT	ECN Capable Transport
ECN	Explicit Congestion Notification
ECN-E	ECN-Echo
ECN-CT	Explicit Congestion Notification Capable Transport
FCH	Frame Control Header
FDD	Frequency Division Duplex
FM	Fairness Measure
FTP	File Transfer Protocol
GFI	Gini Fairness Index
HARQ	Hybrid Automatic Repeat reQuest
HoL	Head of the Line
HTTP	Hyper Text Transfer Protocol
IE	Information Element
i.i.d	independent and identically distributed
IP	Internet Protocol
JFI	Jain's Fairness Index
JOCP	Joint Optimal Congestion control and Power control
KKT	Karush-Kuhn-Tucker
LOS	Line of Sight

MAC	Multiple Access Control
MMI	Min-Max Index
MPEG	Moving Picture Experts Group
ND	Normalized Deadline
O-DRR	Opportunistic Deficit Round Robin
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
OSI	Open Systems Interconnection
PDU	Packet Data Unit
PF	Proportional Fair
PHY	Physical Layer
PM	Poll-Me
PSD	Power Spectral Density
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RED	Random Early Drop
RR	Round Robin
RTG	Receive Transmission Gap
RTT	Round Trip Time
rTPS	Real Time Polling Services
SAP	Service Access Point
SCFQ	Self-Clocked Fair Queuing
SINR	Signal to Interference and Noise Ratio
SNR	Signal to Noise Ratio
SRED	Stabilized Random Early Drop
SS	Subscriber Station
STFQ	Start-Time Fair Queuing
TBFQ	Token Bank Fair Queuing
TCP	Transmission Control Protocol
TDD	Time Division Duplex

TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
TFI	TCP Fairness Index
TTG	Transmit Transmission Gap
TTO	TCP Timeout
TWUS	TCP Window-aware Uplink Scheduling
UCD	Uplink Channel Descriptor
UDP	User Datagram Protocol
UGS	Unsolicited Grant Service
UL_{MAP}	Uplink Map
VBR	Variable Bit Rate
VoIP	Voice over Internet Protocol
WB (CD)	Channel Dependent Weight-based Scheduler
WB (CI)	Channel Independent Weight-based Scheduler
WCTFI	Worst Case TCP Fairness Index
WFQ	Weighted Fair Queuing
WF^2Q	Worst-case Weighted Fair Queuing
WiFi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
W-PC	Congestion control without Power Control
WRR	Weighted Round Robin

List of Symbols

γ	Path loss exponent
μ_l	Network cost associated with link l
λ_l	Congestion cost associated with link l
τ_i	Equilibrium RTT
δ	Step size
ϕ_i	Ideal share of bandwidth allotted to user i
B	Channel bandwidth of i
b_l	Energy cost associated with link l
c_l	Capacity of link l
$cwnd_i$	Congestion window size associated with user i
$d_i(n)$	Delay counter of user i in frame n
D_i	Number of slots required by user i
dc_i	Scaled deficit counter of user i
DC_i	Deficit counter of user i
FM	Fairness measure
FM_w	Wireless fairness measure
FM_{wc}	Worst case fairness measure
H	Routing matrix
i	User index
k	Polling epoch
l	Link index
l_{active}	Active set
$l_{connect}$	Connected set
l_{sch}	Schedulable set

MI	Modulation index
n	Frame index
N	Total number of users connected in the system
N_0	Noise power spectral density
$N_i(n)$	Number of slots assigned to user i in frame n
N_s	Total number of slots available in a frame for uplink scheduling
PL_i	Size of the HoL packet of user i
PL_{Max}	Maximum length of a packet
P_r	Received power
P_t	Transmitted power
Q_i	Quantum of Service assigned to user i in each round
$R_i(n)$	Rate of transmission associated with user i in frame n
$T_d(i)$	Deadline associated with the packets of user i
T_{dl}	Downlink subframe duration
T_H	Deadline associated with the HoL packet
T_f	Duration of a frame
T_s	Duration of a slot
T_{ul}	Uplink subframe duration
T_{uld}	Uplink delay
$Tx_i(n)$	Amount of data transmitted by user i in frame n
q_i	Total cost associated with the source sink pair i
$U_i(x_i)$	Utility associated with the transmission rate x_i
$W_i(n)$	Weight of user i in frame n
x_i	Rate of transmission of source sink pair i
y_l	Aggregate flow at link l

List of Tables

List of Figures

Chapter 1

Introduction

Today, Internet is the de facto source of ubiquitous information access. With applications ranging from email to web browsing, online transaction to job search, weather forecasting to video broadcasting, Internet has an impact on every aspect of our lives. Communication at any time and any place is becoming the necessity of the day. Wireless networks (cf. Figure 1.1) have emerged as the key communication paradigm. Wireless networks such as IEEE 802.11 [1, 2] based Wireless Fidelity (WiFi), IEEE 802.16 [3, 4] based Worldwide Interoperability for Microwave Access (WiMAX) [5], 3rd Generation Partnership Project Long Term Evolution (3GPP LTE) [6], and 3rd Generation Partnership Project 2 (3GPP2) [7] have been defined to provide high quality services. Advances in wireless technology have fuelled an increase in demand for higher data rates and ever-widening customer base. To support the ever increasing demand, efficient solutions that provide high throughput, low delay and fairness guarantee to the applications are required. Moreover, the location dependent time varying wireless channel and limited wireless network resources also influence the design process. The location dependent nature of wireless channel is due to path loss and shadowing, whereas the time varying nature of wireless channel is due to multipath fading and mobility.

In order to meet the applications' requirements while addressing wireless channel constraints, the wireless network resources need to be efficiently allocated. This necessitates efficient resource allocation schemes to be designed. Due to the time varying nature of wireless channel, we can use the information available at the Physical (PHY) and Transport layers for designing resource allocation schemes at the Medium Access Control (MAC) layer to achieve greater end-to-end performance. This approach is known as *cross-layer resources al-*

location [8–13]. This thesis primarily focusses on some such cross-layer resource allocation techniques.

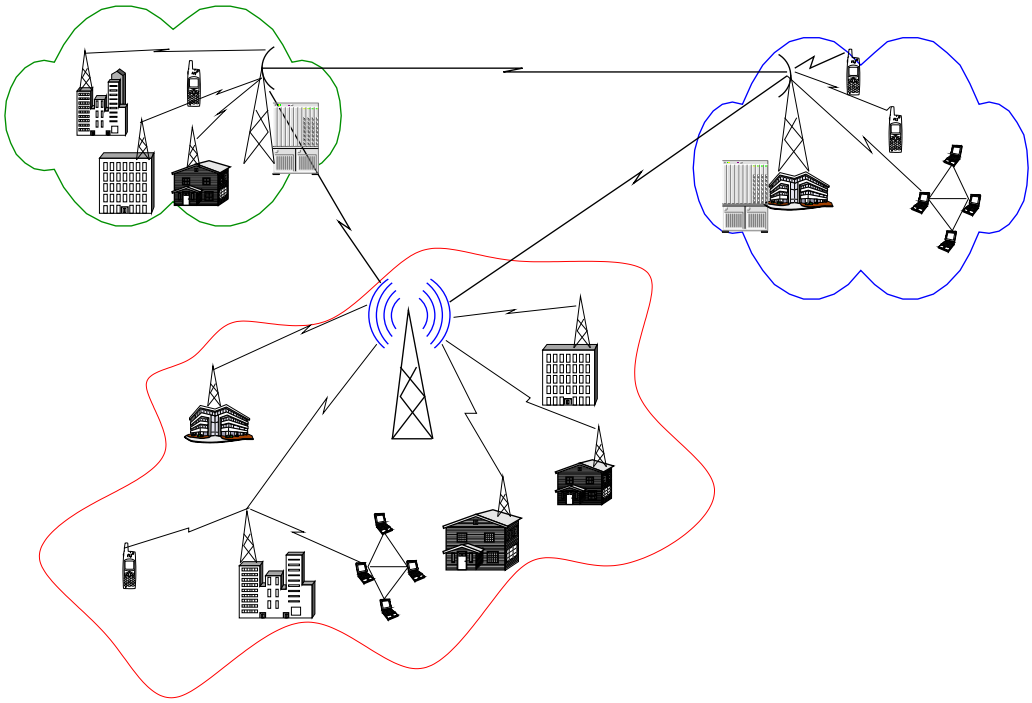


Figure 1.1: A Generic Wireless Network

1.1 Resource Allocation in Wireless Networks: Need for Cross-layer Design

In a multiuser packet switched access network, multiple transmissions are coordinated by a scheduler. Schedulers should be designed such that Quality of Services (QoS) requirements of the applications or users are met. In addition to meeting QoS requirements, maximum utilization of the resources involved and fairness in resource allocation are critical to scheduler design. Depending upon the network types, scheduling can be categorized into centralized or distributed scheduling. In distributed scheduling, each user takes its own scheduling decision based on its local information available, whereas in centralized scheduling, a centralized entity makes the scheduling decisions based on the global information of the users. In this thesis, we limit ourselves to the centralized uplink scheduling.

When the rate of transmission and the number of flows in a link is increased, the input

traffic to that link increases. If the aggregate rate of the flows exceeds the link capacity, then it results in congestion in the network. Since buffers at intermediate nodes are of finite size, congestion in the networks leads to packet drops. Moreover, the queuing delay at the buffers results in delay in packet delivery. Therefore, to control congestion the rate of transmission of individual user should be adapted.

Resource allocation for scheduling or congestion control is more involved for wireless networks than that of wired networks. In a wired network, the links are assumed to be reliable and of fixed capacities, whereas in a wireless network, since the channel is time varying and error prone, the links are assumed to be of variable capacities and un-reliable. Thus, scheduling schemes designed for wireless networks should take into account of the nature of wireless channel, while at the same time ensuring fairness and maximizing resource utilization.

In wired networks, since the links are reliable and of fixed capacities, packet drop is mainly due to congestion in the network. In wireless networks, wireless channel characteristics and interference also contribute to packet drops. Typical Transmission Control Protocol (TCP) based congestion control techniques used for wired networks treat packet drops as an indication of congestion, whereas it is not appropriate in wireless networks. Congestion control in wireless networks can be achieved in two ways: (1) Modification of TCP congestion control mechanism, taking the nature of wireless channel and network model into consideration; and (2) Introducing power control along with congestion control. Since TCP is a popular protocol, modification of TCP to alleviate packet drops due to interference and channel characteristics is not advisable. Therefore, in this thesis, we concentrate on the second approach. Specifically, power control along with congestion control is studied to address the effect of wireless channel and interference.

Since we are dealing with resource allocation schemes, characteristics of resources involved needs to be investigated. Like any other network, wireless networks have a set of resources, such as bandwidth, energy, channel codes, rate and time. Since these resources are limited, resource allocation schemes should maximize the utilization of these resources. Though, all of these resources are of importance in wireless networks, we concentrate primarily on the following two types of resources.

- **Wireless Channel/Bandwidth**: It is the most important resource in wireless networks. Since wireless channel is a shared medium, transmission of more than one node at a time

is prohibitive. Therefore, multiple access techniques like Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA) and Code Division Multiple Access (CDMA) have been used to provide opportunities for transmission to the participating users. TDMA divides wireless channel into time slots and then allocate a time slot to a user. Similarly, frequency bands in FDMA and codes in CDMA are allocated to the users for providing opportunities of transmission. As the number of users increases, opportunity of transmission provided by these systems decreases. This requires effective and maximum utilization of wireless channel.

- **Transmission Energy**: The main characteristics of wireless nodes are limited battery or transmission energy. To ensure longer life time in the network, low power transmission is desired. However, to transmit at a higher rate, nodes need to increase their transmission powers. Since increase in transmission power of a node may result in interference to others, we need to decide when to transmit and at what power level, such that both life time of the network is increased and more amount of data is transferred.

In this thesis, we concentrate our investigations on resource allocation in (i) *infrastructure-based multipoint-to-point wireless networks like IEEE 802.16 networks* (centralized networks) and (ii) *infrastructure-less ad-hoc networks* (de-centralized networks). In the following subsections, we discuss the need for cross-layer resource allocation in these two networks in detail.

1.1.1 Scenario I: Infrastructure-based IEEE 802.16 Networks

In infrastructure-based IEEE 802.16 networks (cf. Figure 1.2), the Base Station (BS) allocates resources, such as physical time slots (WirelessMAN-SC) or channels (WirelessMAN-OFDM, WirelessMAN-OFDMA) to the connected Subscriber Stations (SSs) based on their QoS requirements in both uplink and downlink directions. These are centralized (multipoint-to-point) networks. In IEEE 802.16, transmission between the BS and SSs can be achieved in the assigned time slots or channels either by employing fixed modulation or adaptive modulation schemes at the PHY layer. BS, selects the modulation schemes to be employed dynamically based on the channel states and assigns slots/channels based on the QoS requirements of the upper layer. This channel/slot assignment is termed as scheduling and is a function of the MAC layer of the BS. Though the BS performs both uplink ($SS- > BS$) and downlink

($BS->SS$) scheduling, uplink scheduling is far more complicated than downlink scheduling. This is because, the BS does not have the necessary information of all queues and requirements of SSs in the uplink. Instead of employing scheduling in a First Come First Serve (FSFS) basis or scheduling with equal weights, the BS should assign different number of slots/channels based on the requirements of users. Though the IEEE 802.16 standard defines MAC functionality, it does not define scheduling schemes to be employed.

In IEEE 802.16 networks, the main challenge in designing efficient solutions is the location dependent time varying nature of wireless channel. In order to address the time varying nature of wireless channel, Opportunistic scheduling schemes have been designed at the Medium Access Control (MAC) layers of Open Systems Interconnection (OSI) stack [13–15]. In Opportunistic scheduling, broadly, user with the best channel condition is scheduled and thus multi-user diversity is exploited. This improves the system performance considerably. Opportunistic scheduling schemes are designed for various QoS objectives such as throughput optimality (stability) [16, 17], delay [18, 19] and fairness [20, 21]. These schemes are cross-layer schemes involving Physical (PHY) and MAC layers. However, these schemes have two key limitations: (i) mostly, these schemes are designed for downlink scheduling, in which the packet arrival information and queue state information is available at the scheduler, and (ii) these schemes do not account for higher layers. Indeed, it has been shown that guaranteeing QoS at the MAC layer is not sufficient to guarantee the same at the higher layer [22, 23], e.g., it has been shown that significant variations in instantaneous individual user throughput may result, even if proportional fair scheduler is employed at the MAC layer to provide long-term fairness [22, 24]. Thus, the impact of the Transport layer on the overall performance issue requires further investigation. Since our objective is to provide QoS in terms of high throughput, low delay and fairness guarantee to the applications (both real-time and best effort) in the uplink of multipoint-to-point IEEE 802.16 networks, the characteristics of higher layers (not only PHY and MAC) should also be considered while designing these systems.

1.1.2 Scenario II: Infrastructure-less Wireless Ad-hoc Networks

In infrastructure-less wireless ad-hoc networks (cf. Figure 1.3), nodes are self-configurable and are connected by wireless links. These are de-centralized networks. The communication between nodes is either single-hop or multi-hop in nature. Multi-hop networking improves the

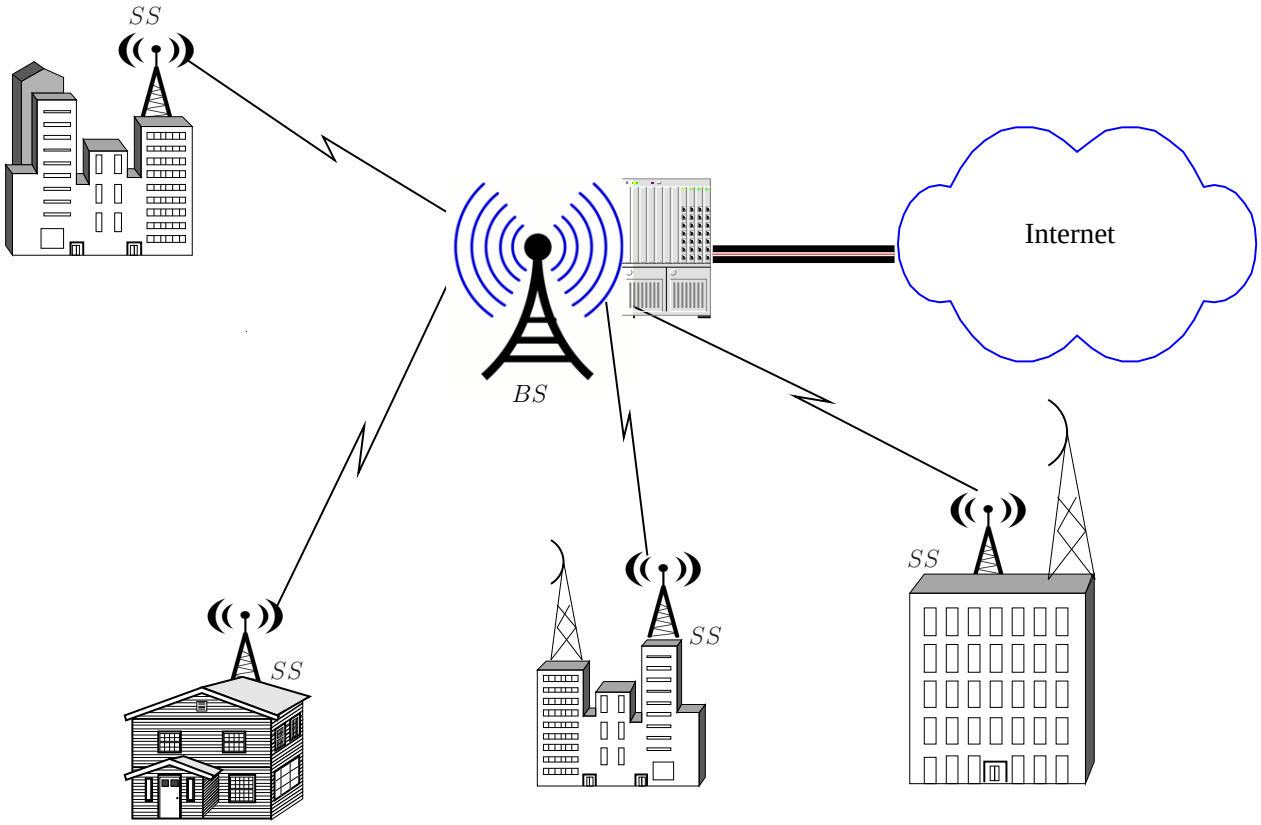


Figure 1.2: Infrastructure-based IEEE 802.16 Networks

power efficiency of the network, as intermediate nodes help in relaying/forwarding traffic to the final destination. The ad-hoc network is advantageous in terms of deployment and maintenance but at the cost of complex time varying and broadcasting wireless links. Multiple hops in an ad-hoc network leads to scalability [25] problem. Due to the de-centralized nature, transmission of one node may interfere with the transmission of another nearby node. In addition, limited energy of wireless nodes also pose challenges in designing schemes to provide higher throughput.

For the best effort services, TCP-based congestion control technique plays a significant role while designing these systems. This is because TCP treats packet drops as an indication of congestion. However, in wireless networks packet drops can occur not only because of congestion in the network, but also due to interference and wireless channel characteristics. Instead of modifying TCP, we propose a channel aware congestion control scheme. In this scheme, congestion in a link can be controlled not only by the TCP congestion control mechanism, but also controlled by increasing transmission power in that link. In addition, power transmission in an un-congested link should be decreased.

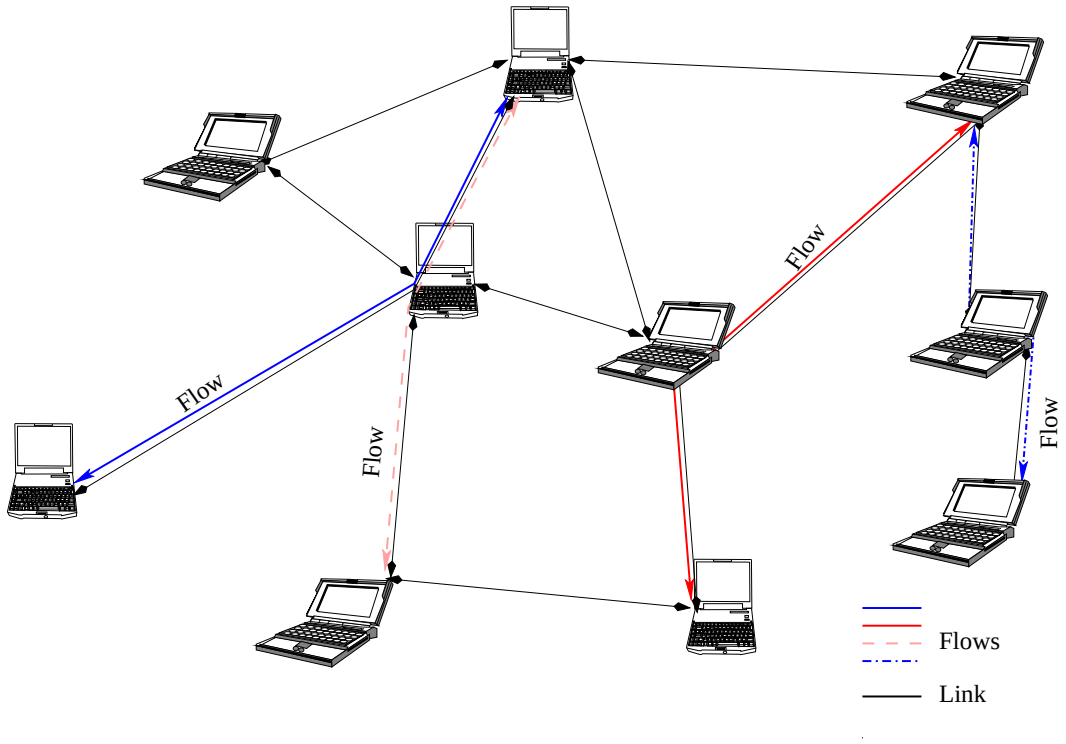


Figure 1.3: Infrastructure-less Wireless Ad-hoc Networks

So far in this chapter, we have discussed wireless resources and the need for cross-layer resource allocation. In the next section, we analyze the gaps in existing research on wireless resources that the present thesis attempts to address.

1.2 Motivation and Contributions of the Thesis

We consider resource allocation techniques using scheduling (for both real-time and best effort services) and congestion control (for best effort services) in wireless networks. Since the Transport layer and wireless channel have significant impact on the system performance, we propose cross-layer resource allocation schemes that take Transport layer information into account in addition to PHY and MAC layer information. In this thesis, we focus only on (I) *Channel and Transport Layer aware Scheduling in Infrastructure-based IEEE 802.16 Networks* and (II) *Channel aware Congestion Control in Infrastructure-less Ad-hoc Networks.*

Part I: *Channel and Transport Layer aware Scheduling in Infrastructure-based IEEE 802.16 Networks*

To study the performance of current implementation of IEEE 802.16 networks, we conduct various experiments in the current deployment of IEEE 802.16 network of one of the

largest telecom service providers in India[1]. We investigate the performance of TCP-based and real-time applications both in the laboratory test-bed setup and in the live-network setup. We observe from these experiments that (i) the throughput of TCP-based application as compared to real-time application is poor, (ii) the channel utilization of TCP-based application is less as compared to that of real-time applications, (for (i) and (ii), we compare systems carrying TCP flows only with systems carrying User Datagram Protocol (UDP) flows only) and (iii) throughput of TCP-based application suffers in the presence of real-time applications (when both TCP and UDP flows run simultaneously). In addition, we observe that the proprietary Weight-based (WB) scheduling scheme implemented in the IEEE 802.16 deployed network of this service provider does not guarantee any scheduling delay to the applications. Since the packets of real-time applications are associated with scheduling deadlines, there is a need to design scheduling schemes that provide delay guarantees. This motivates us to further investigate scheduling schemes which are specific to real-time and TCP-based applications.

IEEE 802.16 standard does not prescribe any particular scheduling scheme, and thus, network elements are permitted to implement their own scheduling algorithms at the BS for both uplink and downlink. We note that the requirements of uplink and downlink flows are different. In the downlink of IEEE 802.16, the BS has knowledge of the queues assigned to each SS, the arrival time of each packet and the individual channel condition of each SS. Hence, the BS can employ a scheduler similar to that of traditional wired networks like Weighted Fair Queuing (WFQ) [26], Self-Clocked Fair Queueing (SCFQ) [27], Worst-case Fair Weighted Fair Queuing (WF^2Q) [28]. However, in the uplink scheduling, the BS does not have packet arrival time and queue state information of SSs, rather this information will have to be acquired. Since communicating packet level information has overhead that is proportional to the number of arriving packets, these scheduling schemes are not scalable and hence not suitable for uplink scheduling. From scalability perspective, Round Robin (RR) or its variants are suitable candidates for uplink scheduling. Note that RR does not require any packet level information. Moreover, to increase the system throughput, we need to exploit the idea of Opportunistic scheduling. Therefore, we need to augment RR scheduling to account for opportunistic allocation.

We now consider the impact of the applications on scheduling design. We first consider real-time applications of Real Time Polling Service (*rtPS*) service class of IEEE 802.16 and

[1]Due to confidential reason, the name of the service provider is not mentioned this thesis.

subsequently consider TCP-based applications of Best Effort (*BE*) and Non Real Time Polling Services (*nrtPS*) service class of IEEE 802.16. Since the real-time application requires maximum delay guarantee, each packet is associated with a deadline. A packet is dropped, when not scheduled before its deadline. Therefore, the proposed scheduler should be aware of the deadline associated with each packet. However, RR scheduler does not take packet deadlines into account, thereby causing packet drops due to deadline expiry. Our aim is to modify RR scheduling so as to reduce packet drops. The key challenge is to achieve the required while (i) keeping control overhead at reasonable level and (ii) utilizing features of IEEE 802.16. One of the key features of IEEE 802.16 standard is the request-grant mechanism for scheduling. In request-grant mechanism, each SS communicates its bandwidth "requests" to the BS. Based on the requirements received at the BS, the BS decides bandwidth for each SS and communicates this information to each SS in form of "grants". Requests from SS can be communicated either in a *contention* mode or in a *contention free* a.k.a *polling* mode. We consider polling mode only as the contention mode does not guarantee any access delay. Access delay is defined as the time interval between the packet arrival at the SS and its departure from SS. In polling mode, the BS polls each SS after every k frames, where $k \geq 1$, called the polling interval or *polling epoch*. Since polling operation has control overhead (in terms of number of slots used for polling), frequent polling should be avoided. This suggests that the value of k should be large, so that the control overhead is reduced. However, when k is large, the system is slow in reacting to the user's changing requirements. This may result in packet drops. Therefore, the polling interval should be carefully chosen to strike a balance between the control overhead and packet drops.

Next, we outline our contributions for uplink scheduling of real-time services. We propose a variant of deficit round robin scheduler, which attempts to schedule flows based on deadlines of their packets and at the same time attempts to exploit the channel condition opportunistically. We call this as "Opportunistic Deficit Round Robin (O-DRR)" scheduler. It is a polling based and low complexity scheduling scheme. In the O-DRR scheme, during the request phase, each user communicates the deadline of its Head of the Line (HoL) packet to the BS. Note that only flow level (not packet level) information is provided to the BS for scheduling. BS in turn, assigns slots to each user based on the deadlines and channel conditions of all users. *Our key contribution is to design scheduling mechanism that decides how many slots should be*

assigned to each user given their requirements and channel conditions. Since O-DRR employs no admission control and does not maintain packet level information, strict delay guarantees cannot be provided to each user. Rather, O-DRR only reduces deadline violations than that in RR scheduler by taking the deadlines of HoL packets into account. Hence, we also attempt to provide fairness among the flows, which in turn guarantees that the packet drops for the applications are similar. For fairness, the O-DRR scheduler employs the concept of deficit counter similar to that of Deficit Round Robin (DRR) scheme [29]. In each frame, the BS updates the deficit counter of each active user and determines weights to assign appropriate slots. Further, we formulate an optimal method to determine the polling interval, such that packet drop due to deadline violation is minimized and control overhead is reduced. To evaluate the performance of the proposed scheduling scheme, we compare the performance of O-DRR scheduler with that of Round Robin (RR) scheduler. We demonstrate that O-DRR scheduler achieves lesser packet drops and higher Jain's Fairness Index than those of RR scheduler for fixed packet sized Video traffic and variable packet sized Pareto traffic (Web traffic) at different system loads, fading and polling intervals.

We then consider the design of scheduling schemes for best effort services in the uplink of IEEE 802.16 networks. The best effort service uses TCP. Our aim is to design centralized TCP-aware uplink scheduling that provides high throughput and fairness to uplink flows. Such scheduling should have low control overhead and should be implementable in IEEE 802.16 system with least modification.

TCP-based applications employ window based congestion control technique and use congestion window size ($cwnd$), Round Trip Time (RTT) and TCP timeout to control the rate of transmission. If the scheduler does not account for TCP, then the flows with $cwnd = 1$ and $cwnd = cwnd_{Max}$ may get equal number of slots. For a flow with $cwnd = 1$, slots may be underutilized as its requirement is small, whereas for a flow with $cwnd = cwnd_{Max}$, the number of slots assigned to it may not be adequate to meet its requirement. This results in increase in scheduling delay and may lead to TCP timeouts. Therefore, if the scheduler does not consider the $cwnd$ size of the TCP flows, then TCP throughput suffers. Like throughput, fairness is also an important parameter. Since TCP flow with small RTT increases its $cwnd$ size at a faster rate as compared to the ones with large RTTs, by scheduling the flows with small RTTs may acquire more number of slots as compared to the flows with relatively large RTTs. Therefore, if

the scheduler does not consider RTTs, it may result in unfairness among the flows. In addition, scheduler should provide delay guarantee such that the occurrence of TCP timeout due to delay in scheduling is minimized.

We propose variants of RR scheduler with request-grant mechanism that provide higher throughput and fairness to the uplink flows. Our contributions in detail is as follows: We propose two scheduling mechanisms "TCP Window-aware Uplink Scheduling (TWUS)" and "Deadline-based TCP Window-aware Uplink Scheduling (DTWUS)" for TCP-based applications that belong to $nrtPS$ and *BE* service class of IEEE 802.16. These are polling based Opportunistic scheduling. In TWUS, $cwnd$ size in terms of slot requirement and RTT information are communicated at the time of polling, where as in DTWUS, TCP timeout information along with slot requirement and RTT are communicated to the BS. To avoid the unfairness due to scheduling based only on $cwnd$ information of users and opportunistic condition, we employ deficit counters similar to that in [29]. The idea of a deficit counter is to ensure fairness among the subscriber stations in long term. Since Adaptive Modulation and Coding (AMC) can be used to achieve high spectral efficiency on fading channels, we also exploit AMC for TCP-aware scheduling by extending TWUS and DTWUS with adaptive modulation. Using AMC, the BS can choose a suitable modulation scheme depending on the Signal to Noise Ratio (SNR) of the users so that the overall system capacity is increased.

We compare the performance of TCP-aware schedulers with that of RR scheduler, Weight-based Channel Dependent (WB (CD)) and Weight-based Channel Independent (WB (CI)) scheduler through exhaustive simulations. We demonstrate that TCP-aware schedulers achieve higher throughput and higher Jain's Fairness Index over RR and WB scheduler. We discuss the implementation of TCP-aware schedulers in an IEEE 802.16 network. We also compare the performance of TCP-aware schedulers with fixed modulation and that of TCP-aware schedulers with adaptive modulations. Though, higher rate of transmission is achieved with adaptive modulation, this higher transmission rate of adaptive modulation is not directly reflected in the average TCP throughout. This is due to the fact that the time scale of change of $cwnd$ size is slower than that of the rate of transmission. We also discuss the properties of TCP-aware schedulers and provide an analysis for TCP throughput in IEEE 802.16 and validate this through simulations. For fairness, we observe that the difference of data transmitted by two back logged flows in any interval is bounded by a small constant.

Part II: ***Channel aware Congestion Control in Infrastructure-less Ad-hoc Networks***

In the second part of the thesis, we consider channel aware congestion control scheme in an ad-hoc network. In this, our prime motive is to provide higher TCP throughput and at the same time transmit at an optimum power level. There are some attempts [30–32] in that direction, which have described cross-layer congestion and power control in wireless networks. These schemes have modelled TCP congestion control in wireless networks as a social optimization problem and employ congestion in a link as a cost function. Since the wireless nodes are mostly battery powered, the transmission power in a link should also be considered as a cost function. However, inclusion of cost function for transmission power in a link may lead to convergence problem. Therefore, further investigation on cross-layer based congestion and power control in ad-hoc networks is required. In addition, implementation of the cross-layer based congestion control in the protocol stack requires further investigation.

To address the above mentioned issues, we propose a channel aware congestion control scheme in ad-hoc networks, which employs both congestion and energy cost. We model TCP congestion control in wireless networks as a social optimization problem and decompose the objective function defined for social optimization in wireless ad-hoc networks into two separate objective functions [33]. We solve these separate objective functions iteratively for the equilibrium transmission rates and link costs (both congestion cost and energy cost). The solution of the decomposed objective functions leads to a cross-layer approach involving TCP and the PHY layer, in which power transmission in a link is determined based on the interference and congestion and the rate of transmission is determined based on congestion in the network.

By controlling transmission power along with congestion control, we demonstrate that congestion in the network as well as average transmission power can be minimized effectively. We investigate the convergence of the proposed scheme analytically. Further, we also perform convergence analysis for various step sizes used for the iterative solution and addition or deletion of flows into the network. We also propose an implementation method for the cross-layer congestion control scheme using Explicit Congestion Notification (ECN) [34]. This is a modified ECN approach involving both ECN-Capable Transport (ECT) code points 0 and 1. It uses 2-bit ECE flag as a function of both average queue size and transmission power instead of the usual 1-bit ECN-Echo (ECE) flag.

1.3 Organization of the Thesis

In this section, we outline the organization of the thesis. This thesis is organized into two parts: (I) Channel and Transport Layer aware Scheduling in Infrastructure-based IEEE 802.16 Networks and (II) Channel aware Congestion Control in Infrastructure-less Ad-hoc Networks. Part I of the thesis is further organized into four chapters and Part II of the thesis is organized into two chapters. We present a summary of the thesis in Chapter 8.

Part I: ***Channel and Transport Layer aware Scheduling in Infrastructure-based IEEE 802.16 Networks***

In Chapter 2, we provide a brief account of wireless fading. We discuss IEEE 802.16 standard in brief and explain its PHY and MAC layer functionality. We review various scheduling schemes used in wireless networks and extend this discussion to uplink and downlink scheduling schemes for IEEE 802.16 networks. Further, we describe the experimental setup employed to investigate the performance of an IEEE 802.16 deployed network and discuss the key findings of our experiments. The findings of our experiments motivate us to investigate scheduling schemes which are specific to real-time and TCP-based applications.

In Chapter 3, we propose O-DRR scheduling mechanism for real-time applications of *rtPS* service class of IEEE 802.16. This is a polling based scheduling scheme. In this chapter, we also formulate an optimal method to determine the polling interval, such that packet drop due to deadline violation is minimized and fairness is maintained. We illustrate the O-DRR scheduling algorithm through an example and discuss implementation framework within IEEE 802.16 setting. Further, we compare the performance of O-DRR scheduler with that of Round Robin scheduler at different system loads, different fading, different polling intervals and different traffic types

In Chapter 4, we propose TWUS and DTWUS scheduling mechanisms for TCP-based applications that belong to *nrtPS* and *BE* service class of IEEE 802.16. These are polling based scheduling. We compare the performance of TCP-aware schedulers with that of Round Robin scheduler and Weight-based schedulers (WB (CD) and WB (CI)) through exhaustive simulations.

In Chapter 5, we consider the TCP-aware scheduling schemes with adaptive modulation. We employ adaptive modulation at the PHY layer and modify the scheduling schemes to accommodate variable rate of transmission, such that fairness is maintained and slot utilization is

maximized. We compare the performance of adaptive modulation based TCP-aware schedulers with that of Round Robin scheduler and Weight-based schedulers through exhaustive simulations. We discuss the properties of TCP-aware schedulers in this chapter and provide a theoretical analysis for TCP throughput in IEEE 802.16 and validate this through simulations.

Part II: ***Channel aware Congestion Control in Infrastructure-less Ad-hoc Networks***

In Chapter 6, we discuss various congestion control techniques used in wired networks, and then extend these to wireless networks. We discuss the need for a cross-layer congestion control (channel aware congestion control) technique and review optimization-based congestion control technique for wireless ad-hoc networks. We discuss some of the open problems in cross-layer based congestion control in ad-hoc networks, which motivates us for designing a joint congestion control and power control scheme.

In Chapter 7, we formulate a joint congestion and power control problem for a CDMA ad-hoc network. In this scheme, each participating node determines its optimal transmission power in an iterative manner to support TCP congestion control scheme. We perform various experiments to determine the efficiency of our framework. We investigate the convergence of the proposed scheme analytically and through simulations. We propose an implementation method for the cross-layer congestion control scheme using ECN in this chapter.

In Chapter 8, we provide concluding remarks and discuss directions for future research.

Part I

Channel and Transport Layer aware Scheduling in Infrastructure-based IEEE 802.16 Networks

Chapter 2

Overview of Cross-layer Resource Allocation in IEEE 802.16 WiMAX Systems

As discussed in the previous chapter, the time varying nature of wireless channel poses one of the key challenges in designing efficient systems. As suggested in the literature, cross-layer resource allocation [8–12] including Opportunistic scheduling [13, 15, 17, 35] are the major approaches designed to handle this challenge. In Opportunistic scheduling, user with the best channel condition is scheduled. This improves the overall network performance considerably. In Part I of this thesis, our objective is to exploit the concept of Opportunistic scheduling and propose cross-layer resource allocation schemes specific to IEEE 802.16 based networks. Since the design of cross-layer resource allocation schemes involve various layers of Open Systems Interconnection (OSI) stack, we need to investigate the key issues arising due to such cross-layer allocation. In this chapter, we discuss various design issues related to cross-layer resource allocation in IEEE 802.16 carrying TCP-based (best effort) as well as UDP-based (real-time) applications. Before designing a cross-layer system, we first conduct various experiments in both laboratory test-bed setup and live-network setup to investigate the performance of IEEE 802.16 deployed network. We perform experiments using both TCP and UDP based applications in uplink as well as downlink directions. Our key findings of these experiments are as follows:

- Throughput achieved by the TCP-based applications is low as compared to that of UDP-

based applications for similar channel states.

- Slot utilization of TCP-based applications is less as compared to that of UDP-based applications.
- Moreover, throughput of TCP-based applications suffers in the presence of UDP-based applications.

From the experimentation, we also observe that the scheduling schemes implemented in the IEEE 802.16 deployed network does not guarantee any access delay to the applications. Access delay is defined as the time interval between the packet arrival at the SS and its departure from SS. Since the packets of real-time applications are associated with deadlines, there is also a need to design scheduling schemes for such kind of applications. The results of our experimentation motivate us to design different scheduling schemes for TCP and UDP applications with the following objectives:

- A scheduling scheme designed for TCP-based applications should maximize the channel utilization and at the same time improve the system performance.
- The scheduling scheme designed for real-time applications should consider deadline associated with the packets.

The above objectives form the basis of our investigation in the subsequent chapters of Part I of this thesis. This chapter is organized as follows. In Section 2.1, we begin with a brief account of wireless channel characteristics. In Section 2.2, we discuss IEEE 802.16 standard in brief and explain the PHY and MAC layer functionality of an IEEE 802.16 based network. We review various uplink and downlink scheduling schemes proposed in the literature for IEEE 802.16 networks in Section 2.3. In Section 2.4, we describe fairness issues related to scheduling. In Section 2.5, we describe the experimental setup employed to investigate the performance of an IEEE 802.16 deployed network and discuss the key findings of our experiments.

2.1 Fading in Wireless Channel

In a wireless network, signal travelling through free space undergoes absorption, reflection, refraction, diffraction and scattering, etc., resulting in variation in received signal strength in a

random manner. This phenomena of random variation of received signal is termed as *fading*. Fading in general can be categorized into: *Large Scale Fading* and *Small Scale Fading*.

- *Large Scale Fading*: Large scale fading is mainly caused due to path loss or attenuation depending on the distance between the transmitter and receiver, and due to shadowing. Large scale fading is typically frequency independent [36, 37].
- *Small Scale Fading*: Small scale fading is mainly caused due to the multipath propagation of the transmitted signal. Since the relative motion between the transmitter and receiver and/or movement of the reflecting objects in the path results in random path lengths, the phase and amplitudes of different multipath components are random. The random changes in amplitudes and phases of the multipath signals result in random change in the received signal, and is termed as small scale fading. Small scale fading can be further categorized into: *flat fading* and *frequency selective fading* [36–38].

2.1.1 Large Scale Fading

The attenuation in signal strength at the receiver due to large scale fading can be expressed as:

$$P_r = P_t d^{-\gamma}, \tag{2.1}$$

where γ is known as the path loss exponent, P_r and P_t are the received and transmitted powers respectively. The path loss exponent γ depends upon the terrain and the environment, typically taking values in the range of 2 to 4. In practice, γ = 2 is used for propagation in free space, and γ = 4 is used for propagation in a relatively lossy environment. In other environments, such as buildings and stadiums, the path loss exponent can reach values in the range of 4 to 6. In addition to the attenuation due to distance, the received signal also undergoes random fluctuations due to the absorption, reflection, refraction and diffraction by the obstacles between the transmitter and the receiver. This random fluctuation in the received signal can be expressed as:

$$P_r = P_t d^{-\gamma} \chi, \tag{2.2}$$

where χ represents *shadowing* random loss. It has been suggested in the literature (Chapter 2 of [39]) that the random variable χ can be modelled as a Log-normal random variable, i.e.,

$$\chi = 10^{\frac{x}{10}}, \text{ where } x \sim N(0, \sigma). \tag{2.3}$$

$N(0, \sigma)$ is a Gaussian distribution (normal) with mean 0 and standard deviation σ. Typical value of σ are in the range of 4-12 dB. The expected value of χ can be expressed as:

$$E(\chi) = \exp\left(\frac{\varrho^2\sigma^2}{2}\right), \tag{2.4}$$

where $\varrho = \frac{\ln 10}{10}$.

2.1.2 Small Scale Fading

Instead of receiving the signal over one Line of Sight (LOS) path, the wireless receiver receives a number of reflected and scattered waves. Since the received signals have varying amplitudes and path lengths with random phases, the instantaneous received power is also random. In literature, this kind of time varying channel has been modelled as a tapped delay line filter with finite number of taps (Chapter 2 of [40]). In this thesis, we limit ourselves to flat fading channels that can be modelled using a single-tap filter with the tape gain modelled as a zero mean complex Gaussian random variable. Let σ_f^2 denote the variance of the tap gain. The magnitude of channel gain is a Rayleigh random variable with probability density function expressed as:

$$f_{\mathcal{H}}(h) = \frac{h}{\sigma_f^2}\exp\left(\frac{-h^2}{2\sigma_f^2}\right),\ h \geq 0, \tag{2.5}$$

and, the squared magnitude $\mathcal{Y}_m \stackrel{\Delta}{=} |\mathcal{H}_m|^2$ is an exponentially distributed random variable with probability density function expressed as:

$$f_{\mathcal{Y}}(y) = \frac{1}{2\sigma_f^2}\exp\left(\frac{-y}{2\sigma_f^2}\right),\ y \geq 0, \tag{2.6}$$

where $2\sigma_f^2$ is the average received signal power and the mean of the exponential random variable. This model is called Rayleigh fading model. After incorporating path loss, Log-normal shadowing and Rayleigh fading, we can express the expected signal power received at a distance d as:

$$\begin{aligned} P_r &= P_t d^{-\gamma}\beta\exp\left(\frac{\varrho^2\sigma^2}{2}\right) \\ &= P_t d^{-\gamma}\beta\exp\left(\frac{(\sigma\ln 10)^2}{200}\right), \end{aligned} \tag{2.7}$$

where $\beta = 2\sigma_f^2$, is the mean of exponential distribution.

The objective of the first part of the thesis is to propose resource allocation schemes for IEEE 802.16 networks. Thus, a brief overview of the IEEE 802.16 is presented in the next section in order to highlight the basic protocol involved in the standard.

2.2 Overview of IEEE 802.16 Standard

Due to the recent technological developments, Broadband Wireless Access (BWA) [3,41] based services turn out to be advantageous than the traditional wired services in terms of fast deployment, flexible architecture, scalability, nomadic access and low cost. BWA systems are expected to support Quality of Service (QoS) for real time applications, such-as Video Conferencing, Video Streaming and Voice-over-IP (VoIP). IEEE 802.16-2004 [3], is a fixed BWA standard for both multipoint-to-point and mesh mode of operation. The standard prescribes WirelessMAN-SC air interface in 10-66 GHz bands based on a single-carrier modulation scheme and WirelessMAN-OFDM, WirelessMAN-OFDMA air interfaces in the band of 2-11 GHz. Along with the fixed BWA, mobile BWA is also supported through the IEEE 802.16e-2005 [4] amendment. In this thesis, we limit our discussions to multipoint-to-point networks based on fixed IEEE 802.16-2004 standard.

IEEE 802.16 standard describes Physical (PHY) and Medium Access Control (MAC) layers of the OSI protocol stack. The Physical layer supports fixed as well as adaptive modulation techniques in both uplink as well as downlink directions. The MAC layer consists of three major sublayers - *service specific Convergence Sublayer* (CS), *Common Part Sublayer* (CPS), and *Privacy Sublayer*. In Figure 2.1, we illustrate the basic IEEE 802.16 protocol structure with the various MAC sublayers.

IEEE 802.16 system architecture consists of two logical entities: Base Station (BS) and Subscriber Station (SS). As per the standard, both SS and BS have instances of IEEE 802.16 PHY and MAC in addition to the support functions. In a multipoint-to-point architecture, BS and SS operate in master-slave relationship. In Figure 1.2, we illustrate a typical deployment of multipoint-to-point IEEE 802.16 network. The basic functions of SSs and BS are as follows:

SS Functions

1. Performs initial ranging, power control, maintenance ranging, etc.

2. Identify the BS to which it needs to be connected. Acquires clock synchronization and

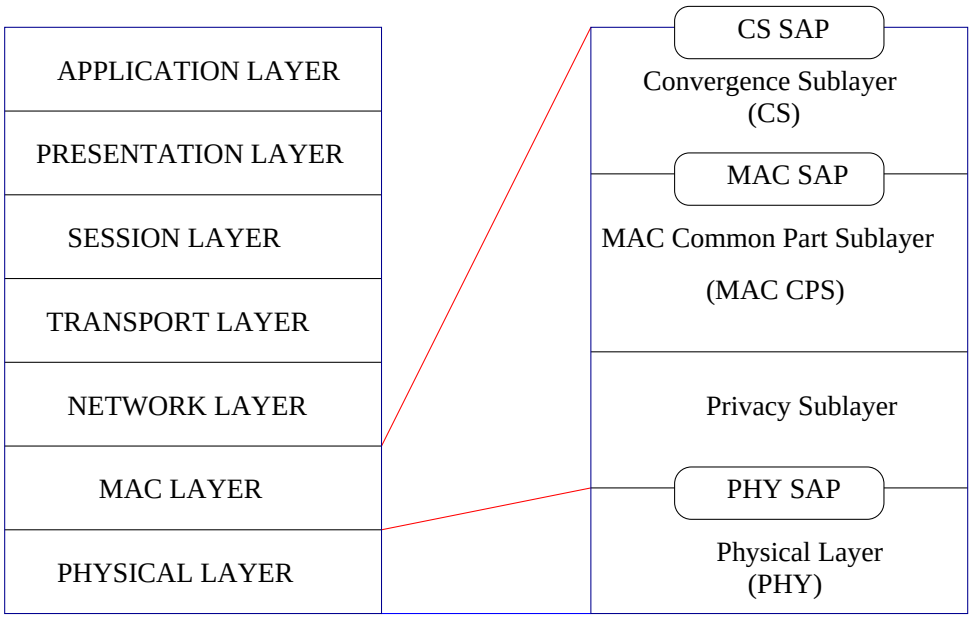


Figure 2.1: Protocol Structure in IEEE 802.16

obtains PHY and MAC parameters from the BS.

3. Establishes a connection with the BS.

4. Generates bandwidth requests to be communicated to the BS for each flow passing through it and schedules each flow based on the bandwidth received from the BS and the flow's requirement.

BS Functions

1. Provides support for ranging, clock synchronization, power control and admission control, etc.

2. Determines the PHY and MAC parameters for all SSs.

3. Performs centralized scheduling both for the uplink as well as downlink flows.

In the subsequent sections, we discuss the operation of IEEE 802.16 standard in detail.

2.2.1 Physical Layer Overview

The standard prescribes WirelessMAN-SC air interface in 10-66 GHz bands based on a single-carrier modulation scheme and WirelessMAN-OFDM, WirelessMAN-OFDMA air interfaces

in the band of 2-11 GHz. In this section, we discuss the basic air interface WirelessMAN-SC in detail.

Since the WirelessMAN-SC air interface operates in the range of 10-66 GHz, Line of Sight (LOS) becomes a practical necessity due to the propagation characteristics. At this frequency range, directional antennas can be used and hence the multipath propagation can be made negligible. The wireless channels between the BS and SSs are logically divided into two different channels; *downlink channel* and *uplink channel*. The downlink channel is a broadcast channel (Time Division Multiplexing (TDM)) and is used by the BS for transmitting data/packets and control information in the downlink direction. The uplink channel is a time-shared channel which is a combination of Time-Division Multiple Access (TDMA) and Demand Assignment Multiple-Access (DAMA). SSs transmit in the designated slots of the uplink channel assigned by the BS.

IEEE 802.16 standard supports both Time Division Duplex (TDD) and Frequency Division Duplex (FDD) mode of operation. Even though the configuration of TDD and FDD systems are different, the basic frame structure supporting adaptive burst profiles is similar in both TDD and FDD. In both TDD and FDD mode of operations, time is divided into frames. The WirelessMAN-SC PHY allows three different frame durations: 0.5 msec, 1 msec and 2 msec. Each frame in turn is composed of a fixed number of slots of equal duration. The number of slots assigned for various purposes (registration, contention, guard, or user traffic) is controlled by the BS and may vary over time for optimal performance. In TDD, each frame is further divided into *uplink subframe* and *downlink subframe*, whereas in FDD, simultaneous transmission in both uplink and downlink are possible in different frequency bands, resulting in no further sub-division of the frame. In this thesis, we discuss TDD based IEEE 802.16 network. Figure 2.2 illustrates the frame structure used by TDD mode of IEEE 802.16. The downlink subframe starts with a preamble, which is used mainly for synchronization and equalization. This is followed by a Frame Control Header (FCH) that contains Downlink Map (DL_{MAP}), Uplink Map (UL_{MAP}), Downlink Channel Descriptor (DCD), and Uplink Channel Descriptor (UCD). DL_{MAP} defines the usage of downlink intervals for a burst mode PHY, whereas UL_{MAP} defines the uplink usage in terms of the offset of the burst, which may refer to the uplink subframe of the current frame or the future frame, depending upon a specific PHY implementation. DCD specifies the downlink PHY characteristics and contains information such as frame duration code

and downlink burst profiles, etc.. UCD specifies the uplink PHY characteristics and contains uplink burst profiles that define uplink interval usage codes and associated PHY characteristics. It also contains the back-off parameters to be used during contention in the uplink. Both DCD and UCD are transmitted by the BS at periodic intervals.

The standard supports both fixed as well as adaptive modulation schemes. In the uplink direction, it supports mandatory Quadrature Phase Shift Keying (QPSK) and optional Quadrature Amplitude Modulation (QAM) schemes (both 16-QAM and 64-QAM), whereas in the downlink, it supports mandatory QPSK and 16-QAM, and optional 64-QAM modulation schemes. Though the standard defines maximum baud rate and modulation schemes to be used for WirelessMAN-SC interface, it does not specify the Signal to Noise Ratio (SNR) thresholds for choosing different modulation schemes to be used.

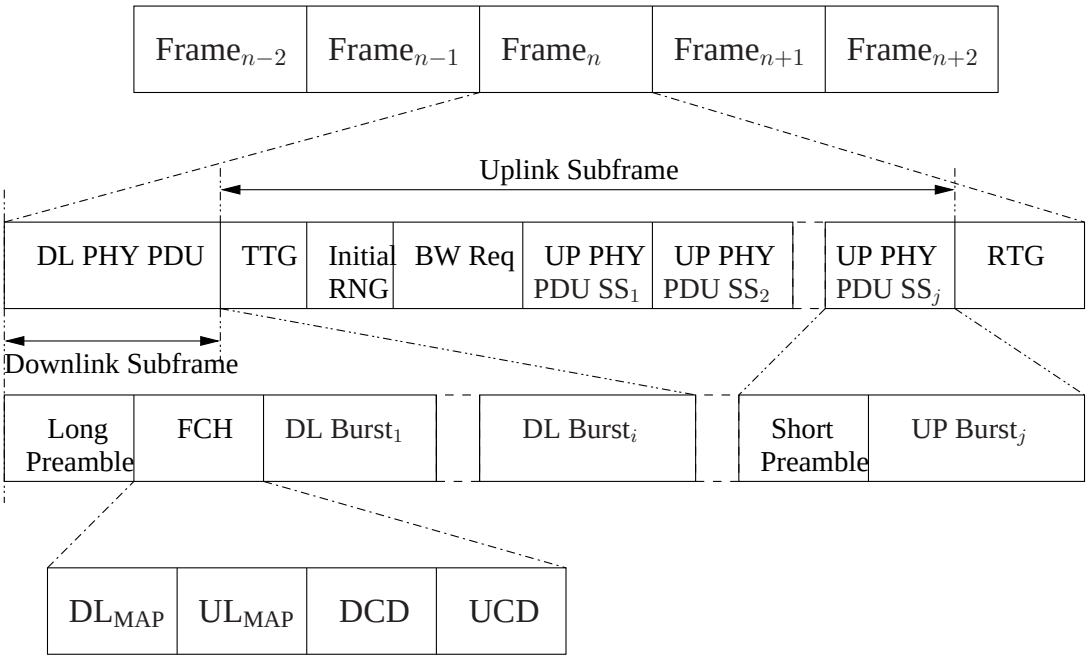


Figure 2.2: Frame Structure in IEEE 802.16

2.2.2 MAC Layer Overview

The primary task of the MAC layer is to provide an interface between the higher layer and the Physical layer. It provides interface to a variety of higher-layer protocols, such as Asynchronous Transfer Mode (ATM), Time Division Multiplexing (TDM) voice, Ethernet, Internet Protocol (IP), etc. It is a connection-oriented MAC, in which each traffic flow between a SS

and the BS can be identified by an unique Connection ID (CID). It supports both connection oriented services such as TCP-based applications and connection less services such as UDP-based applications, and maps each connection oriented and connection less service/application to a MAC layer connection. For the downlink flows, the BS allocates bandwidth to each SS, without involving the SS. However, in the uplink, the BS employs a request-grant mechanism to allocate bandwidth to all users. Note that both uplink as well as downlink scheduling is being performed at the BS. For the uplink scheduling, each SS needs to communicate its requirements (which is known as *requests*) either for each flow or the aggregate requirements of all flows passing through it. The request messages can be sent either in stand-alone mode or in piggybacking mode. Moreover, the bandwidth-requests (BW-requests) can be either incremental or aggregate.

Bandwidth-requests of SSs are normally communicated in two modes: a *contention mode* and a *contention-free mode* (polling). In the contention mode, SSs send BW-requests during a contention period of the uplink subframe. Grants from the BS are communicated to SSs during control slots in the downlink subframe. Contention is resolved using an exponential back-off strategy, and the grants thus communicated are used to schedule data either in the uplink subframe corresponding to the ongoing frame or the next one. In the contention-free mode, the BS polls each SS, and a SS replies with its BW-requests to the BS. IEEE 802.16 MAC supports three types of polling: *Unicast*, *Multicast* and *Broadcast*. In Unicast polling, each SS is polled individually and the BS allocates a request Information Element (IE) directed to the basic CID of the particular SS in the UL_{MAP}. In unicast polling, the polling interval must be such that the delay requirements of the various classes of traffic/services can be met. Multicast polling and broadcast polling are used when sufficient bandwidth is not available to individually poll many active SSs.

After receiving the requests either in polling mode or in contention mode from each SS, the BS assigns bandwidth to each SS in an aggregate manner. The bandwidth allocation details are communicated by the BS in the UL_{MAP} of the frame (cf. Figure 2.2). Based on the requirements of its individual flows and the bandwidth received, each SS assigns bandwidths to individual flows. It is a local decision performed at the SS.

2.2.3 MAC Layer Quality of Service

As discussed before, the connection-oriented MAC of IEEE 802.16 assigns unique Connection IDs (CID) to each traffic flow between SSs and the BS. Further, it defines four kinds of services to which each CID is mapped into. These services are: Unsolicited Grant Service (*UGS*), Real Time Polling Service (*rtPS*), Non Real Time Polling Service (*nrtPS*) and Best Effort (*BE*) service, which we discuss in brief here:

1. *UGS* is designed to provide real-time services, which generate fixed-size data packets on a periodic basis, such as telephony data (E1/T1), VoIP without silence suppression and Constant Bit Rate (CBR) flows. These services do not use any contention requests for BW-request. BS simply allocates fixed-size data grants at periodic intervals to the SSs having *UGS* flows. Explicit BW-requests are issued by the SS for this purpose. The mandatory QoS service-flow parameters for this service are Maximum Sustained Traffic Rate, Maximum Latency, Tolerated Jitter, and Request/Transmission Policy. To reduce the overhead of polling, the BS does not poll SSs with currently active UGS connections If a SS with an active UGS connection needs to request bandwidth for a non-UGS connection, it will set the Poll-Me (PM) bit of a MAC packet used for the UGS connection. This indicates that the corresponding SS needs to be polled. Once the BS detects the request, it starts the individual polling.

2. *rtPS* is designed to support real-time traffic, which is either fixed or variable in both packet size as well as in data rate (real time Variable Bit Rate (rt-VBR)) on a periodic basis like Moving Picture Experts Group (MPEG) video. This is a polling-based service and requires more request overhead than *UGS*. *rtPS* flows are polled at a rate that is frequent enough to meet the delay requirements of the service flows regardless of network load. The mandatory QoS service-flow parameters for this service are Maximum Sustained Traffic Rate, Maximum Latency, and Request/Transmission Policy. To meet the requirements of each $rtPS$ flow, the BS provides periodic unicast polling opportunities. The request/transmission policy of $rtPS$ flow is set, such that SS are prohibited from using any contention requests for these flows.

3. *nrtPS* is used for non real-time traffic that requires variable-size Data Grant Bursts on a regular basis, such as the File Transfer Protocol (FTP). It is a delay tolerant service with

mandatory QoS service-flow parameters like: Minimum Reserved Traffic Rate, Maximum Sustained Traffic Rate, Traffic Priority, and Request/Transmission Policy. This is a polling based service, with the polling being either periodic or non-periodic. The request/transmission policy of $nrtPS$ flows is set, such that SS can also use contention requests along with unicast polling.

4. *BE* Service is intended to provide efficient service to best-effort traffic, such as normal Internet traffic (HyperText Transfer Protocol (HTTP) traffic) with no QoS guarantee. The mandatory service flow parameters for this service are: Maximum Sustained Traffic Rate, Traffic Priority, and Request/Transmission Policy. Applications that belong to BE service do not require any minimum service level and therefore can be handled on the basis of bandwidth availability. The requests/transmission policy of BE flows is set, such that both polling and contention mechanism for bandwidth allocation scheme can be used.

We have so far discussed modeling of channel fading and an overview of IEEE 802.16 standard. We now present a review of representative approaches in literature for scheduling in wired and wireless networks, which is further extended to an analysis of uplink and downlink scheduling for IEEE 802.16 networks.

2.3 Scheduling at the MAC layer of IEEE 802.16

Many scheduling algorithms have been proposed for wired networks [42–44]. These algorithms can be categorized into: (i) *input-driven scheduling*, and (ii) *independent scheduling*. Since scheduling schemes such as Weighted Fair Queuing (WFQ) [26], Self-Clocked Fair Queueing (SCFQ) [27], Worst-case Fair Weighted Fair Queuing (WF^{2}Q) [28], Start-Time Fair Queuing (STFQ), and Deficit Round Robin (DRR) [29] require information such as either packet arrival time or queue size from the users before scheduling, these schemes can be categorized into input-driven scheduling. Other scheduling schemes such as Round Robin (RR), Weighted Round Robin (WRR), and Proportional Fair (PF) schedulers do not require any information from the users while scheduling and hence are categorized as independent scheduling.

Since the nature of wireless channel is different from that of wired network, implementing the above scheduling schemes in wireless networks directly as an extension of wired networks is not appropriate. Various scheduling schemes such as Fair Scheduling [45], Distributed Fair

Scheduling [46], Channel State Dependent Round Robin (CSD-RR) [47], Token Bank Fair Queuing (TBFQ) for Wireless Multimedia Services [48] have been proposed in the literature specifically for wireless networks. Specifically, in IEEE 802.16, the request-grant mechanism, connection oriented MAC, and guaranteed QoS, etc., determine whether a scheduling scheme can be implemented or not. In addition, the requirements of the uplink and downlink also play a vital role while selecting a scheduling scheme.

IEEE 802.16 network elements are permitted to implement their own scheduling algorithms at the BS for both uplink and downlink. However, as discussed in Chapter 1, the requirements of uplink and downlink flows are different. In the downlink of IEEE 802.16, the BS has knowledge of the queues assigned to each SS, the arrival time of each packet and the individual channel condition of each SS. Hence, the BS can employ a scheduler similar to that of traditional wired networks like Weighted Fair Queuing (WFQ) [26], Self-Clocked Fair Queueing (SCFQ) [27], Worst-case Fair Weighted Fair Queuing (WF^{2}Q) [28]. However, in the uplink scheduling, the BS does not have packet arrival time and queue state information of SSs. Since the traditional scheduling schemes like WFQ, SCFQ and WF^{2}Q require the knowledge of packet arrival and queue size at each SS, these schemes are not suitable for uplink scheduling. Instead, variants of Round Robin (RR) schedulers are the suitable candidates for uplink scheduling. We now discuss some of the scheduling schemes that have been proposed in the literature for IEEE 802.16 network.

Most of the existing schedulers for IEEE 802.16 networks have been designed for *rtPS* and *nrtPS* services rather than for *BE* services. In [49, 50], the authors have analyzed the QoS support at the MAC layer by providing differentiated services to applications such as VoIP and web services. They have used Weighted Round Robin (WRR) for uplink and Deficit Round Robin (DRR) [29] for downlink scheduling. Scheduling based on dynamic weights of the IEEE 802.16 flows have also been proposed in the literature [51–53]. In [51], the authors determine the weights of various flows based on the ratio of average data rate of the individual flows to the average aggregate data rate. [52] determines the weights based on the size of bandwidth requests, whereas [53] determines the weights of the individual flows based on the minimum reserved rate. Scheduling based on the delay requirements of *rtPS* and *nrtPS* services have also been proposed in the literature [54, 55]. In [54], the authors propose a Delay Threshold Priority Queuing (DTPQ) scheduling scheme, which determines urgency of *rtPS* flows based on the de-

lay of the Head of the Line (HoL) packets and a fixed delay threshold. This scheme is based on the tradeoff of the packet loss rate of *rtPS* flows with average throughput of *nrtPS* flows. Instead of fixing the delay threshold, the authors also introduce adaptive delay threshold-based priority queuing in [55]. It considers both urgency and channel state information while scheduling *rtPS* and *nrtPS* flows. Instead of using strict priority based scheduling, [56, 57] propose Deficit Fair Priority Queue (DFPQ) based scheduling schemes. [56] uses a *deficit counter* to maintain the maximum allowable bandwidth for each service flow. Based on the value of the *deficit counter*, it decides the priority of scheduling of each flow. In [57], the authors have exploited the use of *deficit counter* for inter-class scheduling in both IEEE 802.16 multipoint-to-point and mesh network.

In [58], the authors have proposed a Token Bank Fair Queuing (TBFQ) [48] based scheduler for the downlink flows of an IEEE 802.16 network. It considers location dependent channel errors while scheduling and employs credit behavior of a flow to determine a priority index required for scheduling. Though this scheme provides fairness, it does not guarantee any delay while scheduling. In [59], the authors propose an adaptive queue aware uplink bandwidth allocations scheme for *rtPS* and *nrtPS* services. The bandwidth allocation is adjusted dynamically according to the variations in traffic load and/or the channel quality. In [60], the authors have proposed a downlink scheduling scheme which considers delay requirement for various classes of services. In this article, uniform preference matrices, such as long-term bandwidth utilization has been used to determine the priority of each queue irrespective of the service class. Since the requirement of each service class varies significantly, scheduling based on uniform preference matrices and delay are not appropriate for IEEE 802.16 networks. Researchers have also exploited the Opportunistic scheduling [13] in IEEE 802.16 networks. Though the Opportunistic scheduling improves aggregate capacity of the network, performance of TCP-based application is degraded due to variable rate and delay, leading to unfairness among the flows. Instead of scheduling uplink and downlink flows separately with two different kinds of schedulers at the BS, authors of [61] have proposed a scheduling algorithm which works both for uplink and downlink flows simultaneously. Further, the authors have claimed that by doing so, they achieve dynamic uplink/downlink resource allocation and better utilization of the channel.

We now consider scheduling of TCP-based applications. In [62], the authors have proposed a contention based TCP-aware uplink scheduling for IEEE 802.16 networks. Further, in [62], SSs do not transmit any BW-requests for scheduling, instead the BS measures the send

rate of each individual flow dynamically and assigns resources based on the measured send rate. This kind of dynamic send rate measurement of all TCP flows at the BS in every frame can lead to scaling problem as the BS has to keep track of the states of all TCP flows along with the SSs requirement. Moreover, the requirement of a TCP flow does not change in every frame, i.e., the requirement is fixed for one RTT. Hence by measuring the send rate in every frame, we may lose more resources than by sending the requirement during polling. Moreover, the scheme in [62] does not consider the time varying nature of wireless channel, the effect of RTT variation on the requirement, and the effect of TCP timeouts. By assigning resources based on the send rate only, some flows might get starved resulting in frequent TCP congestion window ($cwnd$) drops and throughput degradation.

Since RR scheduling and its variations can be used for the uplink scheduling of an IEEE 802.16 network, we discuss RR and DRR scheduling schemes in the subsequent sections.

2.3.1 Round Robin Scheduler

In Round Robin (RR) scheduling, the scheduler visits all users having non-empty queues in a round robin manner to assign resources. The resources obtained by a user during its round robin service opportunity is proportional to its fair share of bandwidth. Since the length/duration of a round depends upon the total number of users in the system, RR scheduler does not provide any guarantee in scheduling delay. It is a starvation-free scheduler, without assigning any priority to any user during scheduling. RR scheduler does not require packet arrival information for scheduling.

2.3.2 Deficit Round Robin Scheduler

Deficit Round Robin (DRR) scheduler [29, 63] is a modification of RR packet scheduler in which, a quantum Q_i of service is assigned to each user in each round. It assigns ideal share (weight) $\phi_i = \frac{Q_i}{Q}$ to each user i, where $Q = \min_i\{Q_i\}$. Let PL_i be the size of the HoL packet of user i. In the first round, the scheduler schedules the packets of only those users whose quantum size exceeds their HoL packet size, i.e., HoL packet of user i is scheduled if $Q_i \geq PL_i$. It maintains a state variable known as *deficit counter* DC_i for each user i, which keeps a track of the amount of bits transmitted by user i. At the beginning of the first round, the deficit counter of each user is set to zero, i.e., $DC_i(0) = 0$. At the end of the first round,

the scheduler updates the deficit counter of each user by adding the un-utilized quantum, either fully un-utilized, if $PL_i > Q_i$, or partially un-utilized, i.e., $Q_i - PL_i$, to its deficit counter. If user i does not have a packet waiting in its queue, then the deficit counter is reset to zero. This can be expressed as:

$$DC_i(1) = Q_i, \text{ if } PL_i > Q_i,$$
$$DC_i(1) = \max(0, Q_i - PL_i), \text{ if } Q_i \geq PL_i > 0,$$
$$DC_i(1) = 0, \text{ otherwise.}$$

To avoid examining the queue sizes of the users, the DRR algorithm uses an auxiliary list known as *active list*. It is the list of indices of the users that contain one or more packets in their queues. Whenever a packet arrives to a previously empty queue of a user i, user i is added to the end of the active list. For scheduling, the scheduler maintains a round robin pointer known as *DRR pointer* which points the next user in the list for scheduling.

In the next round, the scheduler checks the packet size PL_i and the deficit counter value DC_i. If the sum of deficit counter value and quantum size exceeds the packet size, i.e., $DC_i + Q_i \geq PL_i$, then only the packet is scheduled. Similar to the previous round, the scheduler also updates the deficit counter of each user by adding the un-utilized quantum to its deficit counter and subtracting the size of its HoL packet, which can be expressed as:

$$DC_i(2) = DC_i(1) + Q_i, \text{ if } PL_i > Q_i + DC_i(1),$$
$$DC_i(2) = DC_i(1) + \max(0, DC_i(1) + Q_i - PL_i), \text{ if } DC_i(1) + Q_i \geq PL_i > 0,$$
$$DC_i(2) = 0, \text{ otherwise.}$$

DRR scheduler schedules the HoL packets of each user of the active list and this process continues.

In Figure 2.3, we illustrate the operation of DRR scheduling through an example. We consider four users in this case. The DRR pointer points to the first packet of user 1 (size of 20 unit) in the first case. The quantum assigned for all four users is 40 units. Since the size of the packet corresponding to the DRR pointer (user 1) is less than the quantum size, the scheduler schedules this packet. After scheduling this packet, it subtracts 20 units (equal to packet size) from the DRR counter and points the DRR pointer to the next user (user 2). Since the size of the

first packet of user 2 is 50 units, which is more than the sum of the quantum and deficit counter value for user 2, this packet will not be scheduled. Hence, the scheduler adds the quantum size to the deficit counter and moves the DRR pointer to the next user, i.e., user 3. This process continues in the first round. In the first round, it can schedule first packet of user 1 and user 3 only. However, it will not be able to schedule first packet of user 2 and user 4.

In the second round, the scheduler can schedule the first packet of user 2 and user 4 only. This is because, the sum of the deficit counter and quantum size is 80 units each for user 2 and user 4, which is more than the size of their packets waiting at the HoL (50 units and 70 units respectively). However, in this round, it cannot schedule the HoL packet (second packet) of user 1 and user 3. This is because, the sum of the quantum and deficit counter is 60 units for user 1 and 40 units for user 3, whereas the packet sizes of the HoL packets are 70 units for user 1 and 50 units for user 3.

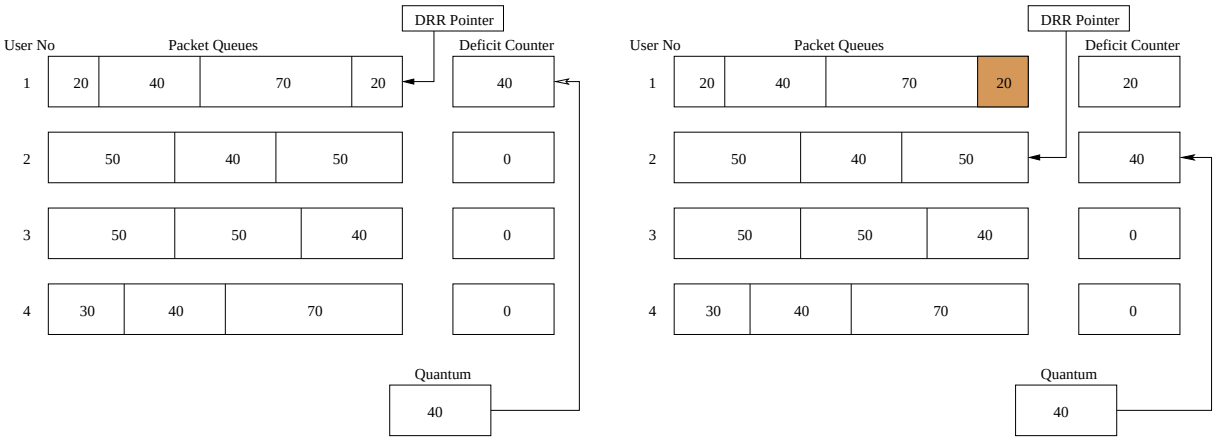


Figure 2.3: Deficit Round Robin Scheduling in a Wired Network

Authors in [29,63] have discussed the fairness properties of the DRR scheduler for a wired network. Similar to RR scheduling, DRR also provides complexity of $O(1)$ provided that the quantum size $Q_i \geq PL_{Max}$, where PL_{Max} is the maximum length of a packet.

For any kind of resource allocation scheme, maximization of resource utilization and fair distribution of resources have been identified as two important goals. In the next section, we describe representative literature in fairness in resource allocation.

2.4 Fairness in Resource Allocation

Fairness is an important parameter required to be considered while designing and operating resource allocation schemes. Depending upon the nature of the system and application involved, the notion of fairness differs. In literature, various methods have been described to measure fairness either in a qualitative manner or in a quantitative manner [29,42,64–66]. In this section, we describe some of the frequently used methods to measure fairness of scheduling.

Based on the time interval over which fairness is measured, the notion of fairness also differs. The resource allocation scheme is long-term fair if:

$$\lim_{m\to\infty} \frac{\sum_{n=1}^{m} Tx_1(n)}{m} = \cdots = \lim_{m\to\infty} \frac{\sum_{n=1}^{m} Tx_i(n)}{m} = \cdots = \lim_{m\to\infty} \frac{\sum_{n=1}^{m} Tx_u(n)}{m}, \tag{2.8}$$

where $Tx_i(n)$ is the amount of resources allocated to user i in the n^{th} frame and m is the number of frames over which fairness is measured. Similarly, we can express short-term fairness as:

$$\frac{\sum_{n=1}^{m} Tx_1(n)}{m} = \cdots = \frac{\sum_{n=1}^{m} Tx_i(n)}{m} = \cdots = \frac{\sum_{n=1}^{m} Tx_u(n)}{m}, \tag{2.9}$$

where m, the number of frames over which fairness is measured is small and finite. Along with short-term or long-term fairness, *fairness measure* has also been used to describe the notion of fairness [29]. It is defined as the maximum difference between the normalized service received by two users over any time interval in which both are backlogged. In other words, we express fairness measure as:

$$FM(t_1,t_2) = \max_{\forall i,j} \left| \left(\frac{Tx_i(t_1,t_2)}{\phi_i} - \frac{Tx_j(t_1,t_2)}{\phi_j} \right) \right|, \forall (t_1,t_2), \tag{2.10}$$

where (t_1,t_2) is the interval over which fairness is measured, ϕ_i and ϕ_j are the share of user i and j respectively and $Tx_i(t_1,t_2)$ and $Tx_j(t_1,t_2)$ are the services received by user i and j respectively. Both users i and j are backlogged during the interval (t_1,t_2). A resource allocation scheme is considered to be fair, if the fairness measure defined above is bounded by a small constant irrespective of the intervals.

Other than short-term/long-term fairness and fairness measure, fairness indices such as Jain's Fairness Index (JFI) [65], Gini Fairness Index (GFI) [64], and Min-Max Index (MMI) [64] have also been used in the literature. Jain's Fairness Index is expressed as:

$$\text{Jain's Fairness Index (JFI)} = \frac{[\sum_{i=1}^{u} x_i]^2}{u \sum_{i=1}^{u} {x_i}^2} \tag{2.11}$$

where x_i is the amount of resource allocated to user i and u is the total number of users. JFI is independent of any time scale and lies between 0 to 1; 0 being unfair and 1 being fair. Even though JFI can be used as a reliable fairness index for comparing the throughput achieved by the contending users, recent studies in [66] have shown the necessity of transport layer fairness, if the applications are TCP-based. As discussed in [66], there is a need to investigate Transport layer fairness as the MAC layer fairness is not sufficient for elastic traffic. It defines two Layer-4 (Transport layer) fairness indices, namely Worst Case TCP Fairness Index (WCTFI) and TCP Fairness Index (TFI). Both WCTFI and TFI are measured for the proposed scheduler in relative to Round Robin scheduler. We define WCTFI and TFI as follows:

Let ψ_i denote the throughput achieved for user i at the Transport layer by the proposed scheduler and let ς_i denote the Transport layer throughput received for user i by the Round Robin scheduler, then

$$\text{WCTFI} = \min_{\forall i} \left[\mathcal{M}\left(\frac{\psi_i}{\varsigma_i}\right) \right], \tag{2.12}$$

where $\mathcal{M}$ is a positive real-valued function defined as:

$$\mathcal{M}(\vartheta) = \begin{cases} \vartheta, & \text{if } 0 \leq \vartheta \leq 1 \\ 1, & \text{otherwise.} \end{cases}$$

If the $WCTFI$ value is 1, then the scheduler is perfectly fair. The authors in [66] define another index TCP Fairness Index (TFI) as:

$$\text{TFI} = \frac{[\sum_{i=1}^{u} \mathcal{M}(\frac{\psi_i}{\varsigma_i})]^2}{u \sum_{i=1}^{u} \mathcal{M}(\frac{\psi_i}{\varsigma_i})^2}. \tag{2.13}$$

TFI captures the relative fairness among the users.

In the next section, we describe experimental studies conducted on one of the IEEE 802.16-2004 deployed networks of a leading telecom operator in India. We also present a summary of our findings from the experimental setup that forms the basis of our investigation in the first part of the thesis.

2.5 Experimental Evaluation of Scheduling Schemes in a Telecom Provider Network

We perform experiments in both laboratory test-bed setup and in live-network setup of an IEEE 802.16 deployed network. In Figure 2.4, we illustrate the laboratory experimental setup, in which two SS modules are connected to one BS module. We emulate the time varying wireless channel characteristics between SSs and the BS using a channel emulator. We connect one application terminal (Laptop) to each of the SS modules and one to the BS module. We employ traffic generators to generate both UDP-based applications (real-time services) and TCP-based applications (best effort services). We use software based traffic monitors to monitor the details, such as sequence numbers, type, packet size, arrival and departure time, etc. We measure the performance of both UDP-based and TCP-based applications in uplink as well as in downlink directions.

In Figure 2.5, we illustrate the live-network setup involving IEEE 802.16 deployed network and the Internet. We connect two terminals as shown in Figure 2.5 - one at the remote end within wireless network and the other on the Internet. In this experimental setup, we also employ traffic generators to generate TCP-based as well as UDP-based applications. We employ software based traffic monitors at both the terminals to monitor the details of packets arrival/departure.

Since we perform various experiments in the IEEE 802.16 deployed network, we illustrate the operation of the "Weight-based Scheduler" - employed by the service provider for uplink and downlink scheduling in this section.

2.5.1 Weight-based Scheduler

Weight-based (WB) scheduler is a proprietary scheduler implemented at the BS to schedule both uplink and downlink traffic. It assigns scheduling priority or weight to each user in a weighted fair scheduling way. If W_i is the weight of user i, then resource allocated (BW_i) to user i is such that the ratio $\frac{BW_i}{W_i}$ is same for all users. Weights of the users can be modified dynamically. For example, in a wireless network, if the rates of transmission of users are different, then the amount of bandwidth assigned among the users should be such that $\frac{BW_i}{\frac{1}{R_i}}$ is same $\forall i$, where R_i is the rate of transmission associated with i^{th} user. This kind of Weight-based

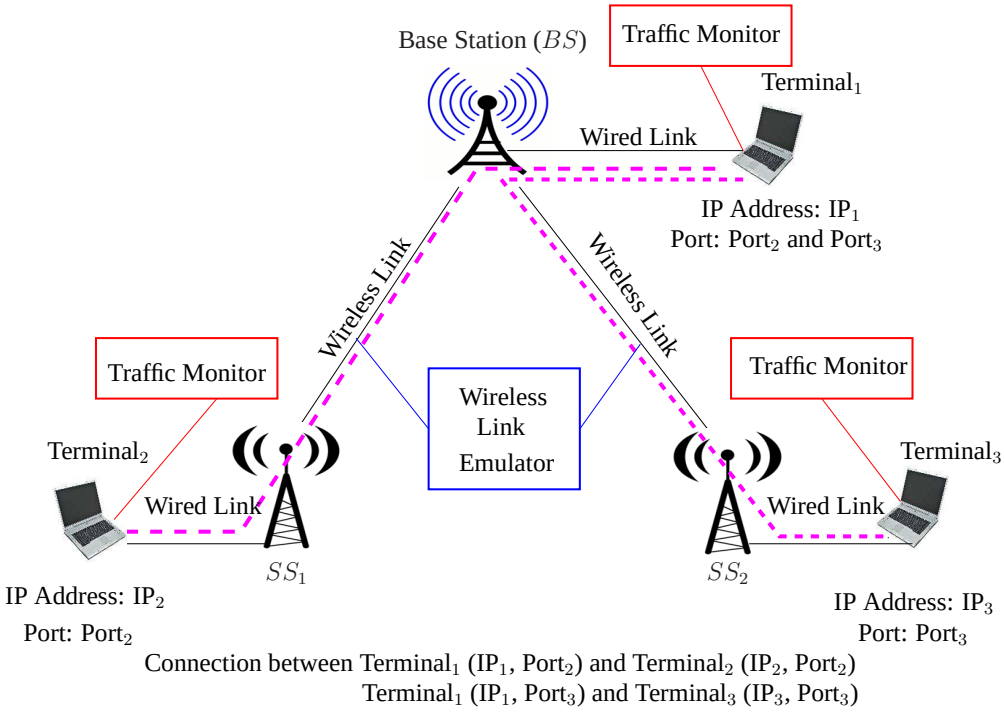


Figure 2.4: Test-bed Experimental Setup: Laboratory Environment

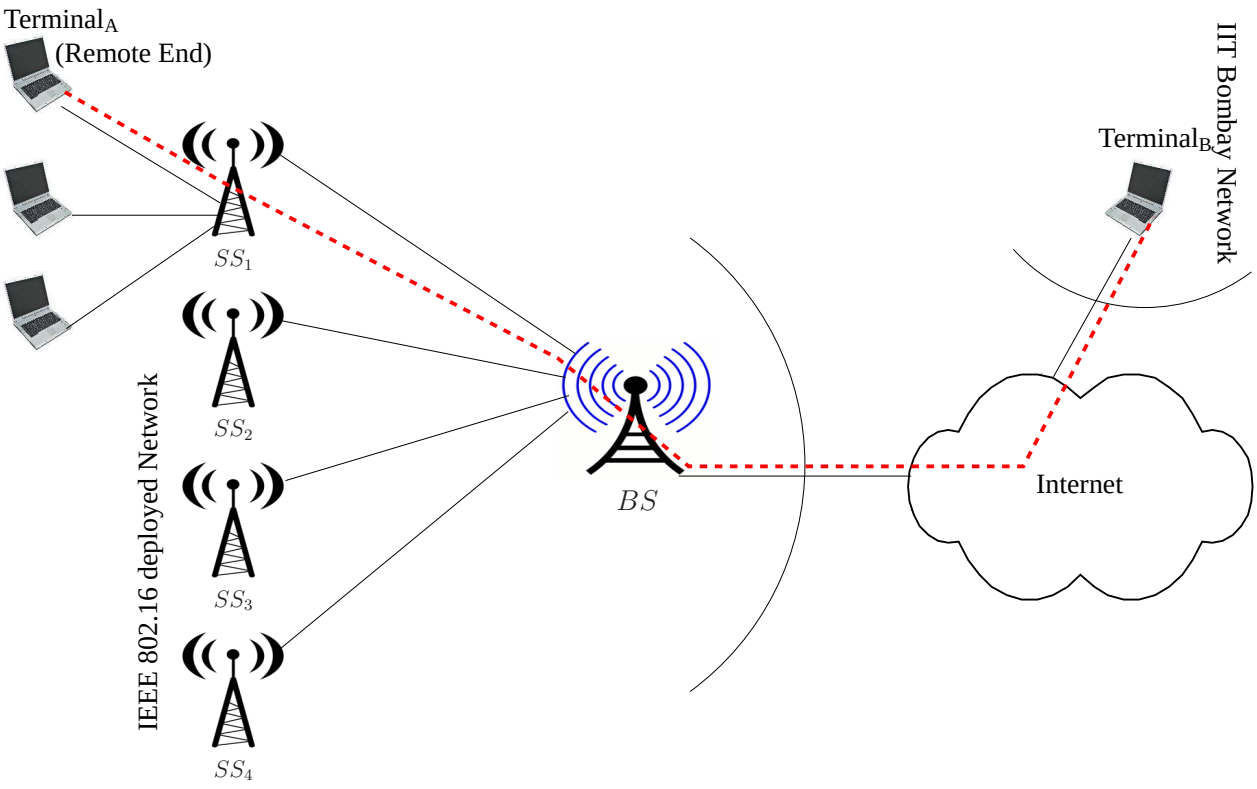


Figure 2.5: Experimental Setup involving Live-network

scheduling is known as Channel Dependent Weight-based scheduling WB (CD). If the variation in the rate of transmission is not considered during scheduling, i.e., slots are assigned only on the requirements, then it is known as Channel Independent Weight-based scheduling WB (CI). The weights of each user in both WB (CI) and WB (CD) scheduling can be interpreted as a function of its requirements. Though, there are provisions for both WB (CD) and WB (CI) scheduling, WB (CD) is used in the implementation.

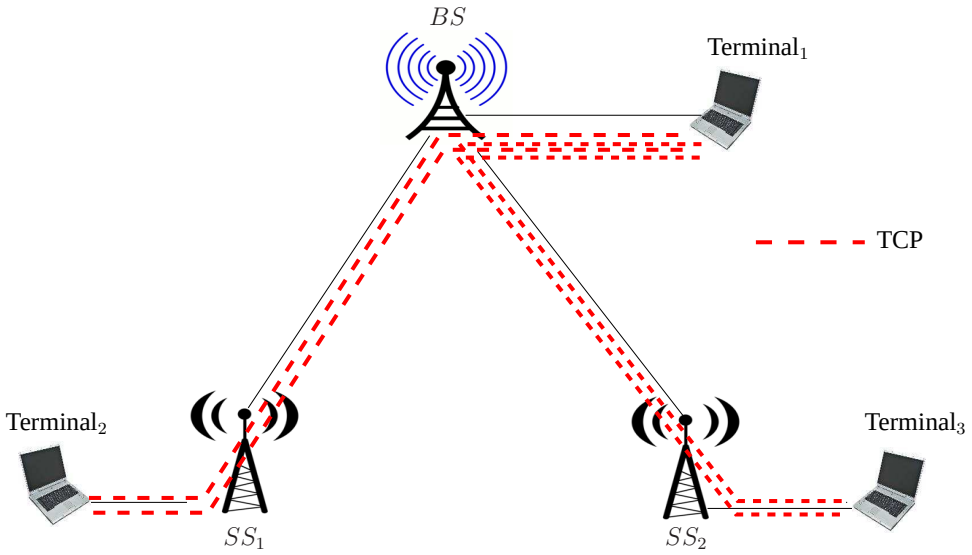
2.5.2 Experiments and Measuring Parameters

Before performing the experiments, we study system parameters of the network. We then describe the experimental setup and the measuring parameters.

PHY and MAC Layer Settings

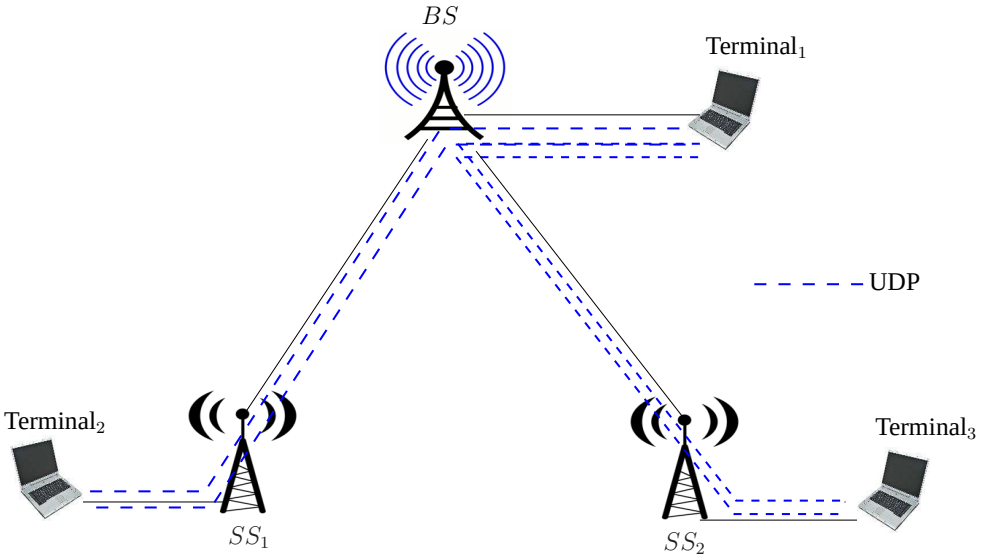
Both the test-bed and the live-network employ short-term averaged Carrier to Interference-plus-Noise Ratio (CINR) based link adaptation algorithm to select modulation schemes and coding. Further to add robustness in the noisy environments, two parameters such as: *protection* and *hysteresis* are utilized. The first parameter "protection" provides a margin in the CINR levels for selecting modulation schemes. The second parameter "hysteresis" prevents continuous modulation switching at modulation thresholds due to CINR fluctuations. For reliable and low BER operation in fast fading and high interference radio environment, high values of protection and hysteresis are desired. The values of protection and hysteresis are set to 4 dBm each. Both the test-bed and live-network setups employ fixed and adaptive modulation schemes. We set the duration for CINR averaging required for selecting modulation schemes to 10 sec. We perform the experiments with and without Automatic Repeat-reQuest (ARQ) set in the system. In Table 2.1, we summarize other important system parameters used in the network setup.

Based on the applications, we categorize the experiments into three types: (i) TCP-based applications, (ii) UDP-based applications, and (iii) mixed traffic (both UDP-based and TCP-based applications). In Figures 2.6 - 2.8, we illustrate the experimental setup for the test-bed experiments involving all three categories. We conduct experiments for a sufficiently large duration of time and perform measurements through the traffic monitors over every ten minutes for each experiment. We repeat the experiments for different combinations of uplink and downlink transmissions. We also conduct similar experiments in the live-network.



TCP Traffic between Terminal$_2$ and Terminal$_1$, Terminal$_3$ and Terminal$_1$

Figure 2.6: Test-bed for Experimental Evaluation: Category I



UDP Traffic between Terminal$_2$ and Terminal$_1$, Terminal$_3$ and Terminal$_1$

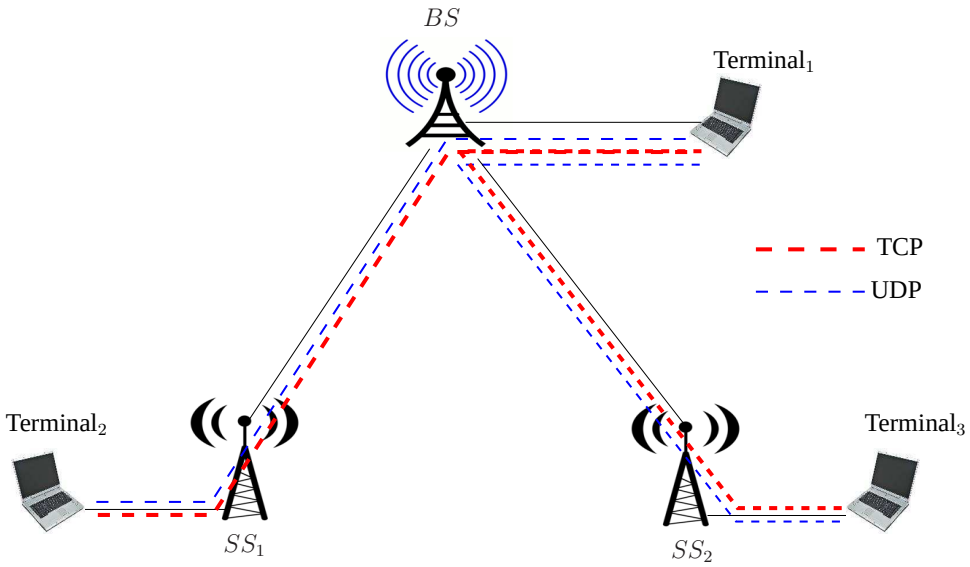
Figure 2.7: Test-bed for Experimental Evaluation: Category II

Table 2.1: Summary of System Parameters

System Parameters	**Value**
Air Interface	OFDM
Duplex Method	TDD
Modulation Schemes	64-QAM 3/4, 64-QAM 2/3, 16-QAM 3/4, 16-QAM 12 QPSK 3/4, QPSK 1/2, BPSK
Frame Duration (T_f)	10 msec
CINR Average Duration	10 sec
$T_{ul} : T_{dl}$	3:2
QoS Priority	Upto 16 classes per SS
Scheduling Scheme	Weight-based Scheduler
ARQ	With and Without ARQ
Hybrid HARQ (HARQ)	No

2.5.3 Experimental Results

In this section, we analyze the performance of both the laboratory test-bed setup and the live-network. From Table 2.2, we observe that the average throughput achieved by the TCP-based applications in test-bed setup is higher than that of the live-network, in both uplink and downlink directions. Moreover, the throughput achieved by the downlink flows is higher than that of the uplink flows, in both live-network as well as in test-bed setup. We also observe that the throughput achieved by the TCP-based applications with ARQ is higher as compared to that achieved without ARQ for similar channel states. To investigate the reason behind the higher throughput achieved in ARQ implemented system, we measure the percentage of packet re-transmission occurred in the test-bed setup with and without ARQ set, which we present in Table 2.3. From Table 2.3, we observe that the percentage of re-transmission drops drastically when ARQ is set in the network. The drop in re-transmission of packets enables the applications to achieve higher throughput when ARQ is set. We also observe that the percentage of re-transmission of packets in live-network is higher than that of the re-transmission occurred in test-bed setup. Moreover, the re-transmission percentage is more for uplink flows than that of downlink flows, when ARQ is not set. However, when the ARQ is set, we observe less re-transmission for the uplink flows than that of downlink flows.



Mixed TCP and UDP Traffic between Terminal_2 and Terminal_1, Terminal_3 and Terminal_1

Figure 2.8: Test-bed for Experimental Evaluation: Category III

Table 2.2: Throughput Achieved by TCP-based Applications (in kbps)

	Without ARQ		**With ARQ**
Type	Test-bed	Live-network	Test-bed
Uplink	408.4	238.1	558.9
Downlink	684.6	400.3	743.1

Table 2.3: Re-Transmission of TCP Packets (in %)

	Without ARQ		**With ARQ**
Type	Test-bed	Live-network	Test-bed
Uplink	20.68	9.93	2.2
Downlink	20.28	6.1	4.18

Table 2.4: Throughput Achieved by UDP-based Applications (in kbps)

	Without ARQ		**With ARQ**
Type	Test-bed	Live-network	Test-bed
Uplink	858.8	905.0	910.6
Downlink	847.7	995.4	952.8

We also compare the throughput achieved by both TCP-based and UDP-based applications. In Table 2.4, we present the throughput achieved by the UDP-based applications for different experimental setups. From Table 2.2 and 2.4, we observe that the throughput achieved by UDP-based applications are substantially higher (more than even 100% in some cases) than that of TCP-based applications for similar channel states, irrespective of the network types. We also observe that even though both TCP and UDP-based applications have same priority for scheduling, the UDP-based applications transmit more packets as compared to that of TCP-based applications, resulting in higher throughput.

Since the Weight-based scheduler used for uplink as well as downlink scheduling does not differentiate TCP and UDP flows, it assigns time slots to both type of flows in the same order. Therefore, by assigning equal number of slots, a TCP flow with a small congestion window ($cwnd$) size will not be able to utilize all the slots assigned to it, if the number of slots assigned to it are more than its requirement (which is a function of $cwnd$ size). On the other hand, when the $cwnd$ size is very large and the number of slots assigned is not sufficient, then it will result in $cwnd$ drop and degradation in throughput. For UDP flows, since the rate of transmission is independent of the number of slots assigned, these kind of flows will be able to transmit at their peak rates, resulting in complete utilization of the slots assigned and higher throughput. Hence, by assigning equal number of slots to both TCP and UDP flows, TCP flows will not be able to utilize the slots assigned to them fully resulting in lesser throughput as compared to the UDP flows.

We also observe the performance of TCP-based applications with sudden change in channel states. In addition, we observe the performance of TCP-based applications in the presence of other UDP and TCP-based applications running simultaneously in different network setups.

In Figure 2.9, we plot a snapshot of throughput achieved by an uplink TCP flow in a

test-bed experiment (Category I). From this figure, we observe that the instantaneous throughput fluctuates between 10 Kbytes/sec to 95 Kbytes/sec. The large fluctuation in instantaneous throughput can be attributed due to the high percentage of re-transmission of TCP packets observed in our experiments. We also observe similar throughput variations for the downlink TCP flows. In Figure 2.10, we plot a snapshot of the throughput achieved by an uplink TCP flow in a live-network experiment. From this figure, we observe that even though the instantaneous throughput fluctuates frequently, the average throughput is almost constant for the entire duration of the experiment. We observe similar variations for the downlink TCP flows also.

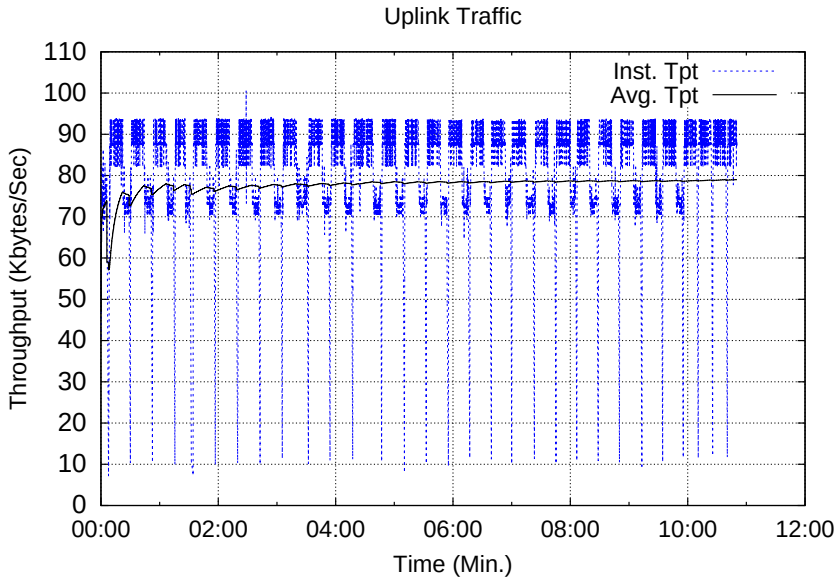


Figure 2.9: Snapshot of Throughput Achieved by a TCP Flow (Uplink, Test-bed, Without ARQ)

In Figure 2.11, we plot the instantaneous throughput achieved by an uplink TCP flow with manual change of the channel state (Category I). We observe that the sudden drop in instantaneous throughput (after 0.8 min, in Figure 2.11) occurs, when we introduce more noise to the channel. We also observe that for a relatively noisy channel the fluctuation in instantaneous throughput is less as compared that to a less noisy channel. In Figure 2.12, we plot the instantaneous throughput achieved by a downlink TCP flow with manual change of the channel state. In this experiment, we manually tune the channel states in every one minute (at 1, 2, 3 min, in Figure 2.12). From this figure, we observe that the instantaneous throughput varies significantly when the channel state changes. Even though the short-term CINR averaging is being used in the system, we still observe fluctuations in throughput. This may be due to the improper scheduling scheme implemented in the system.

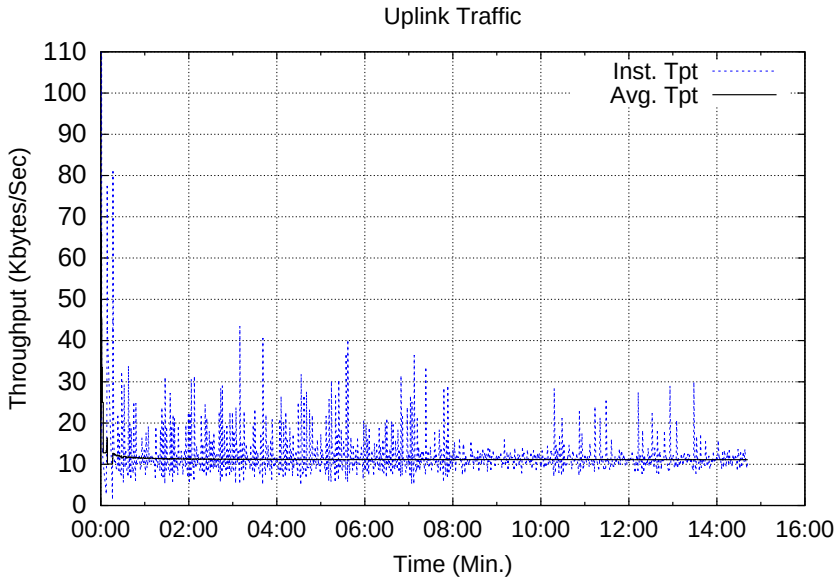


Figure 2.10: Snapshot of Throughput Achieved by a TCP Flow (Uplink, Live-network, Without ARQ)

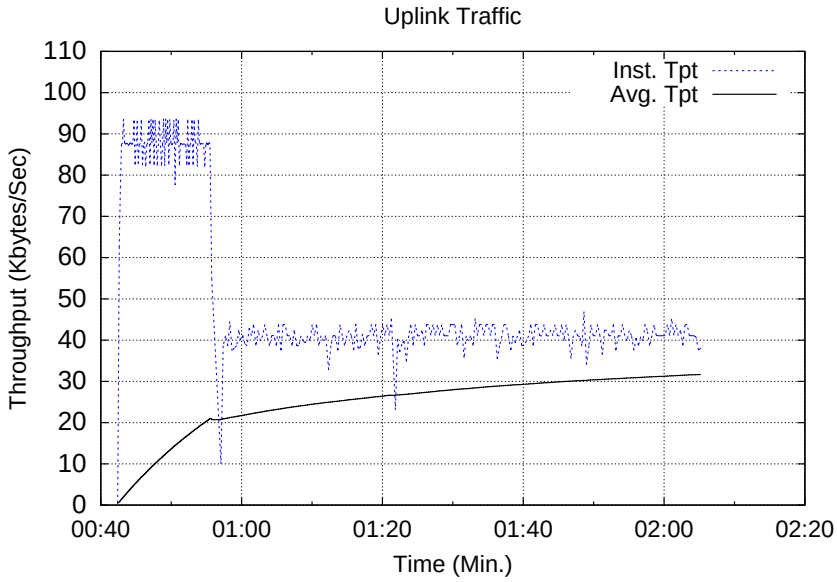


Figure 2.11: Effect of Change in Channel State on TCP Throughput (Uplink, Test-bed, Without ARQ)

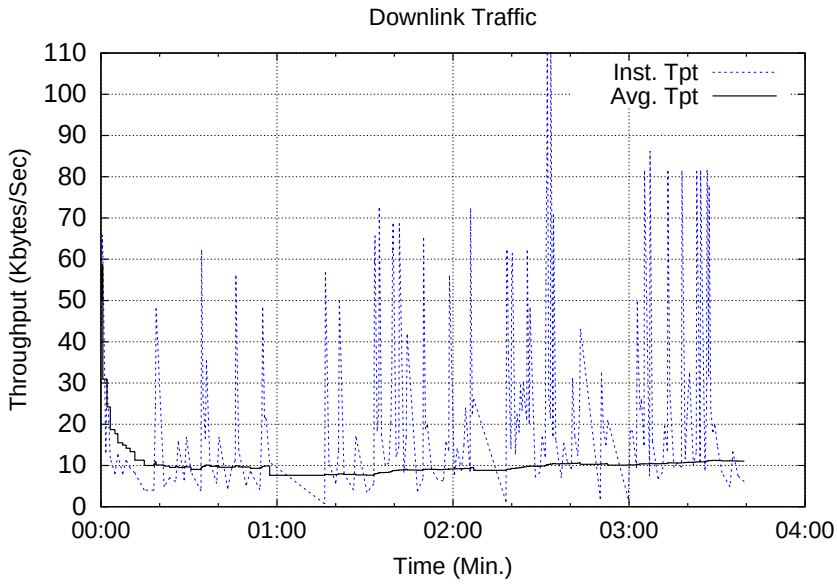


Figure 2.12: Effect of Change in Channel State on TCP Throughput (Downlink, Test-bed, Without ARQ)

We also investigate the effect of a TCP flow on another TCP flow running through the same SS. For this, we conduct experiments with two TCP flows simultaneously running, the second flow starts after 1.30 min of the start of the first flow. From Figures 2.13 - 2.14, we observe that the fluctuation of instantaneous throughput is more, when more than one TCP flows are running. This effect can be described as follows: by assigning equal number of slots to both the flows, the scheduler may force one flow to drop its $cwnd$ size due to insufficient slots assigned to it, even when the slots assigned to the other flow are not being utilized fully.

We also investigate the effect of UDP flows on the performance of TCP flows when both TCP and UDP flows run simultaneously (Category III). For this, we conduct experiments with one TCP and one UDP flow running simultaneously through same SS and stop the UDP flow after one minute. We repeat this experiment both in the uplink and downlink directions. We observe that when both flows run simultaneously, the TCP-flow gets starved and all the slots assigned to one SS is being utilized by the UDP flow only. The TCP flow transmits only when the UDP flow stops. Through Figure 2.15, we demonstrate that the instantaneous throughput of the TCP flow drops to zero just after it starts (initial one minute of this figure) and increases its rate of transmission to its usual state only after one minute (when the UDP flow stops). In Figure 2.16, we plot the instantaneous throughput achieved by a downlink TCP flow in the presence of a downlink UDP flow. Similarly, we also observe that the downlink TCP flows also are affected by the ongoing UDP flows (drop in throughput between 0.50 min to 02.50 min, in Figure 2.16).

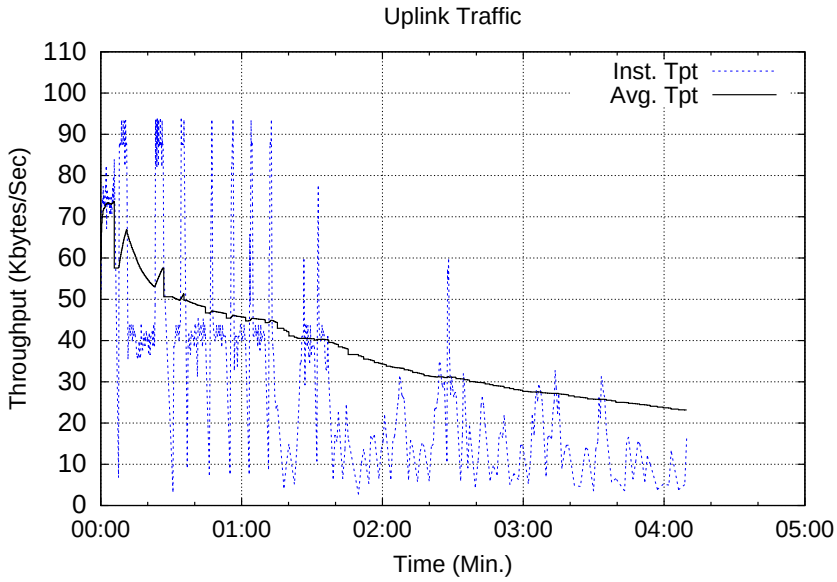


Figure 2.13: Effect of a Parallel TCP Flow - Second Flow Starts at 1.30 minute (Test-bed, Without ARQ)

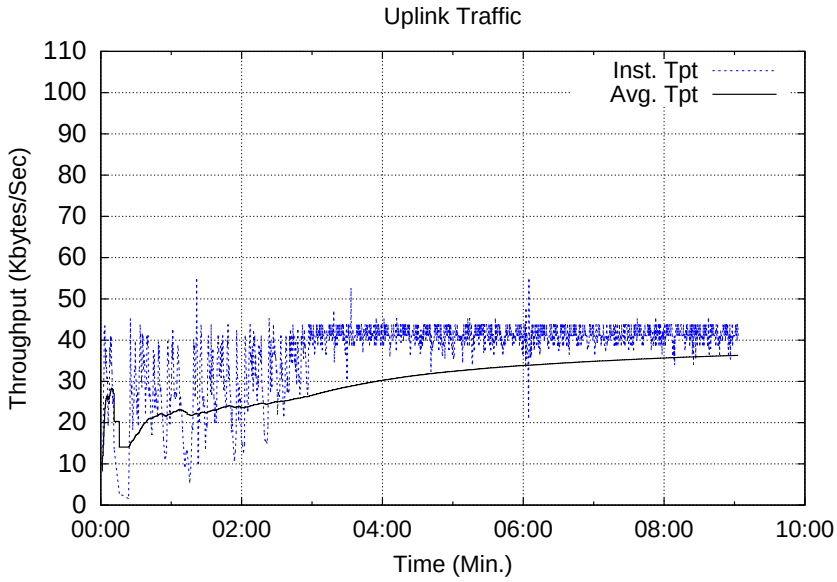


Figure 2.14: Effect of a Parallel TCP Flow - First Flow Stops at 3.00 minute (Test-bed, Without ARQ)

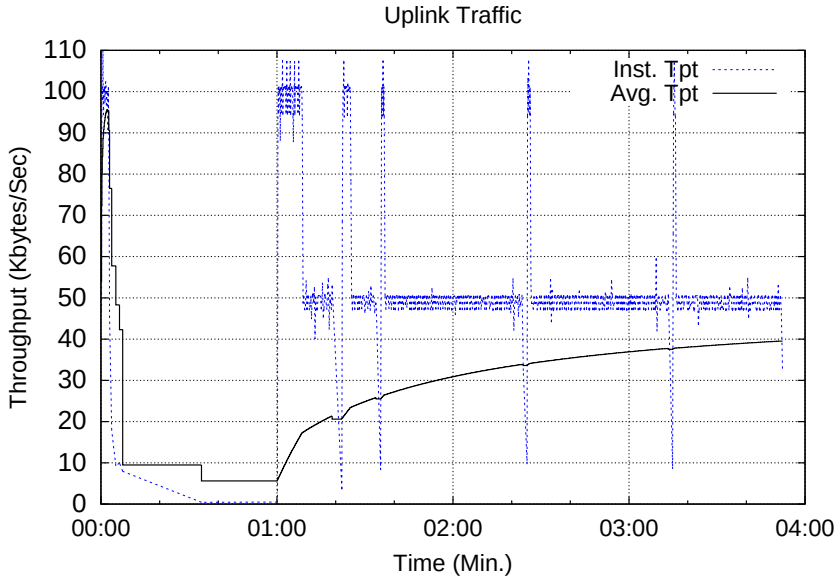


Figure 2.15: Effect of UDP Flow on TCP Flow (Uplink, Test-bed, With ARQ)

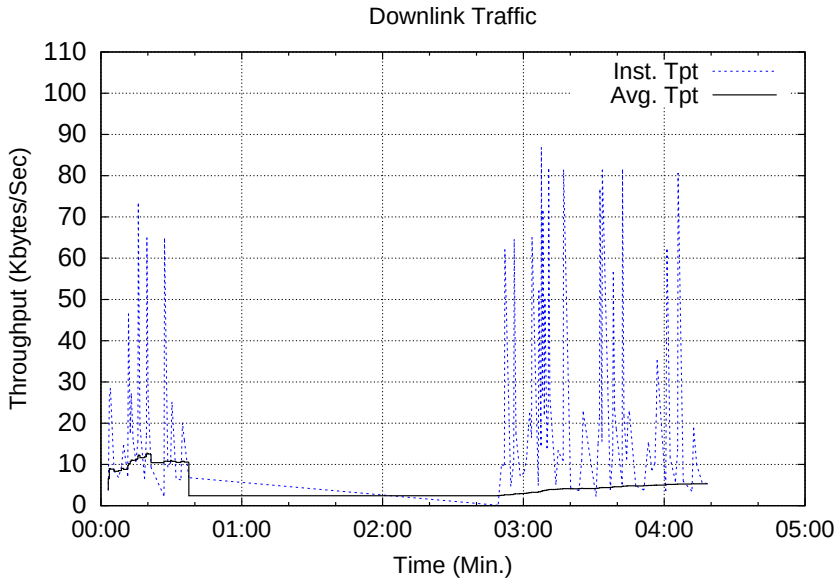


Figure 2.16: Effect of UDP Flow on TCP Flow (Downlink, Test-bed, Without ARQ)

This demonstrates that the version of Weight-based scheduler implemented in the deployment is biased towards UDP flows, resulting in denial of services to TCP flows, both in the uplink and in the downlink directions. Though we have illustrated the results of test-bed experiments, similar results are also valid in live-network setup.

From both test-bed experiments and live-network experiment, we observe that low throughput achieved by TCP-based applications is mainly due to the nature of scheduling. The Weight-based scheduler, which is implemented in the IEEE 802.16-2004 deployed network does not provide any guarantee in terms of delay to TCP traffic, resulting in timeout in most of the cases. It is also biased toward UDP flows. Moreover, it does not assign slots based on one's requirement, resulting in slot under-utilization and throughput degradation. Since it does not guarantee any scheduling delay, packets of real-time services may get dropped at SS due to deadline violation. From these observations, we plan to investigate alternative scheduling schemes which can provide better throughput to TCP-based applications and optimize channel utilization as well as provide better services to real-time applications. We discuss these scheduling schemes in detail in the subsequent chapters.

Chapter 3

Deadline based Fair Uplink Scheduling

In this chapter, we present a novel scheme for cross-layer based uplink scheduling in a multipoint-to-point network. The proposed scheme attempts to balance the worst-case fairness in slot allocation and the deadline requirements of multi-class traffic, while taking the varying nature of wireless channel into account. Though the algorithm presented in this chapter is applicable to any cellular network, we consider the setting of IEEE 802.16 based WiMAX network. As explained in the previous chapter, WiMAX has four different classes. Among these classes, Real Time Polling Service (*rtPS*) meant to support real time applications, has maximum latency or deadline being one of the QoS parameters. Consequently, we assume that each packet has a deadline or maximum delay that a packet can tolerate. At the time of connection set up, users can negotiate their QoS requirements with the base station (BS). During the connection phase, *rtPS* traffic may be required to notify the BS of its current resource requirement. We consider Demand Assignment Multiple Access-Time Division Multiple Access (DAMA-TDMA), which adapts to the demands of multiple users by dynamically assigning time slots to users depending upon their current QoS requirements. Specifically, we assume that the BS polls each subscriber station (SS) and the SS in reply communicates its QoS requirements. The polling interval needs to be chosen such that the twin objectives of meeting packet deadlines and being fair in slot allocation among the SSs may be achieved. Having obtained the optimum polling interval, we propose a mechanism for slot allocation that meets the QoS objectives. Unlike traditional scheduling algorithm like Weighted Fair Queuing (WFQ) [26] and their variants [28] for wireless networks, the proposed scheduling algorithm does not require any fluid-flow assumptions in the background and makes no assumption about the queue size at the SS's.

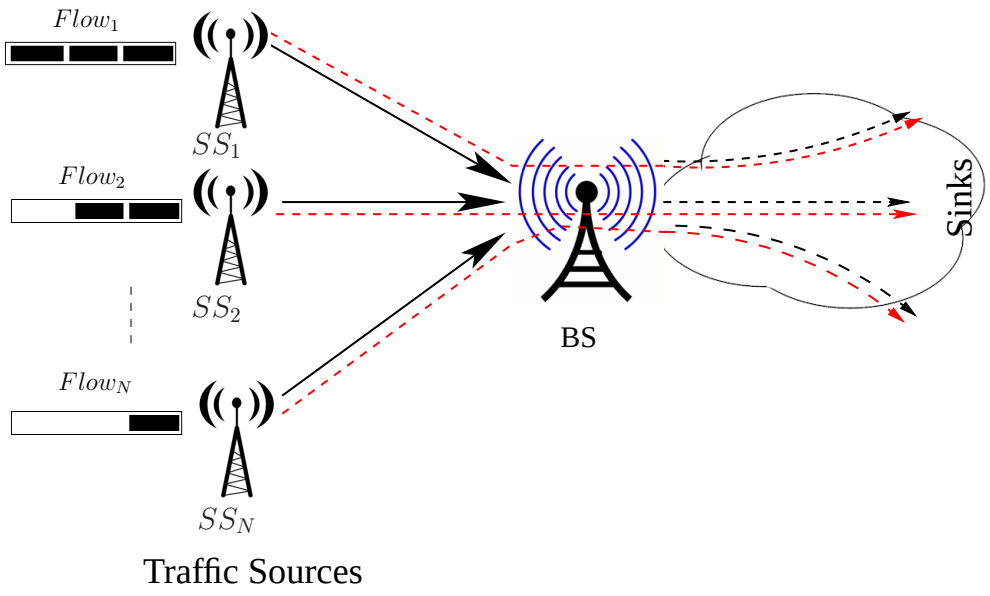


Figure 3.1: Multipoint-to-Point Scenario

In Section 3.1, we discuss the system model and the motivation behind the deadline based fair uplink scheduling. In Section 3.2, we discuss determination of an optimal polling interval and propose the scheduling steps. In Section 3.3, we illustrate our algorithm through an example and discuss implementation framework within IEEE 802.16 setting. In Section 3.4, we discuss the experimental set up and present the results. These results demonstrate the efficiency of the proposed algorithm.

3.1 System Model and Problem Formulation

We consider a multipoint-to-point scenario (as in IEEE 802.16/WiMAX standard [3, 4]) where multiple SSs are connected to a BS as shown in Figure 3.1. BS is the centralized entity responsible for scheduling the flows.

For simplicity, we consider a single flow per SS, even though the proposed algorithm also works for multiple flows per SS. Each packet is associated with a deadline. A packet is dropped, if it is not scheduled before the expiry of the deadline. All packets of a flow have equal deadlines. However, deadlines across the flows may be different. Flows can be classified into multiple classes based on their deadlines. Flows with the same deadline belong to the same class.

Time is divided into frames. Each frame (of duration T_f) in turn is composed of a fixed

number of slots of equal duration of T_s. In Time Division Duplexing (TDD) mode of operation, (as considered in simulations later) each frame is further divided into *uplink subframe* and *downlink subframe*. We assume time varying wireless channel. However, the coherence time of the channel is assumed to be greater than the frame length, i.e., the channel state does not change during a frame duration. The channel state changes from frame to frame according to Rayleigh fading model [37, 38]. We also consider path loss and Log-normal shadowing [67] in modeling channel gain. We assume channel reciprocity, i.e., uplink and downlink channel gains are the same. Further, we assume that the individual channel state information is available at the BS in every frame. Let SNR_i denote the Signal to Noise Ratio (SNR) measured at the BS for the channel between SS_i and the BS. Packets can be successfully received if $SNR_i \geq SNR_{th}$, where SNR_{th} denotes a threshold whose value depends upon the modulation and coding scheme employed at the Physical (PHY) layer and the Bit Error Rate (BER) requirement of the application.

3.1.1 Motivation

The objective of the scheduling algorithm is to schedule flows in every frame by assigning appropriate number of slots to each flow. In the centralized uplink scheduling, SSs are required to communicate their bandwidth requests to the BS. BS, in turn, assigns slots to the SSs. Bandwidth requests can be communicated either in a *contention* mode or in a *contention free* or *polling* mode. Since the contention mode does not guarantee any access delay, we consider polling based scheduling to meet the deadline requirements. In the polling based scheduling, the BS can poll each SS after every k frames, where $k \geq 1$, called the polling interval or polling epoch. Since polling operation has an overhead (in terms of number of slots used for polling), frequent polling should be avoided. This suggests that the value of k should be large. However, a large polling interval can lead to deadline expiry and unfairness among the flows. The unfairness can result because of the fact that if a flow misses polling[1], it does not get scheduled in the entire polling interval. The polling interval should be carefully chosen to strike a balance between the overhead due to polling, packet drops and fairness. In Section 3.2.1, we derive an optimal polling interval.

As discussed in Chapters 1 and 2, traditional scheduling schemes like Weighted Fair Queu-

[1]A flow can miss a polling when either it has a low SNR or an empty queue at the polling instant.

ing (WFQ) [26], Self-Clocked Fair Queueing (SCFQ) [27] and Worst-case Fair Weighted Fair Queuing (WF^{2}Q) [28], etc., cannot be used for the uplink scheduling. This is because in the uplink, communication of packet arrival time from SS to the BS and communication of virtual start time and finish time from the BS to SS for each packet arrived at a SS is not possible. Instead, variants of Round Robin (RR) schedulers are suitable candidates for uplink scheduling. Since the channel state of SSs varies randomly across the frames, a RR scheduler would result in unfairness. Moreover, RR scheduler does not take packet deadlines into account, thereby causing packet drops due to deadline expiry. We, therefore, propose a variant of deficit round robin scheduler, which attempts to schedule flows based on deadlines of their packets and maintains fairness among flows. We term this as "Opportunistic Deficit Round Robin (O-DRR)" Scheduler.

3.2 Opportunistic Deficit Round Robin Scheduling

Opportunistic Deficit Round Robin scheduling (O-DRR) proposed in this chapter is a variant of Deficit Round Robin (DRR) [29] scheduler which is popular in wired network. In DRR, the algorithm maintains a quantum size Q_i and a deficit counter DC_i. The larger the quantum size, larger is the share of bandwidth assigned to a flow. In each round, the algorithm schedules as many packets as possible for flow i with total size less than $Q_i + DC_i$. The packets that are not scheduled account for the deficit in DC_i for the next round. The algorithm is provable to be fair in the long run for any combination of packet sizes [29]. We modify the DRR to take into account the varying nature of the wireless link to satisfy the fairness and the deadline constraint. To increase the system throughput, we exploit the idea of Opportunistic scheduling while designing O-DRR scheduler.

Before discussing the scheduling algorithm, we define the following terms.

- *Connected Set*: The set of SSs that has been admitted into the system through an admission control and connection set up phase is called connected set ($L_{connect}$). Let N be the cardinality of the connected set.

- *Polling Epoch*: Polling epoch is an interval that the BS chooses to poll the connected SSs. In the proposed algorithm, the polling is performed by the BS once after every k frames, i.e., the polling epoch comprises of k frames.

- *Schedulable Set*: A SS is schedulable, if it has a non-empty queue and the SNR of its wireless link to the BS is above a minimum threshold (say, SNR_{th}). The set of such SSs at the beginning of a polling epoch constitutes schedulable set L_{sch}. This set is fixed for one polling epoch and may change dynamically across the polling epochs. Let M be the cardinality of the set.

- *Active Set*: An *active* SS is defined to be one that is schedulable during a given frame of a polling epoch and that was schedulable at the beginning of that epoch. The set of such SS constitutes an active set L_{active}. During a frame of a polling epoch, the BS only schedules traffic from the corresponding active set.

- *Quantum Size*: It is the number of slots that should be assigned to any of the schedulable SS in any frame of a polling epoch. At the beginning of every polling epoch, the BS determines the quantum size as: $Q = \frac{N_s}{M}$, where N_s is the total number of slots available for uplink scheduling. Since M is fixed for a polling epoch, Q is also fixed for a polling epoch.

For each frame in the polling epoch (i.e., for every frame for the next k frames), the BS schedules (using Opportunistic Deficit Round Robin, to be described shortly) the transmissions of the schedulable SSs. Note that the membership of the active set *changes dynamically* from frame to frame during a polling epoch, depending on the state of the channel between the SS and the BS. At the end of k frames, the BS re-determines the states of all of the SSs, and begins the above process over again. The relationship between polling epoch and frame by frame scheduling is shown in Figure 3.2.

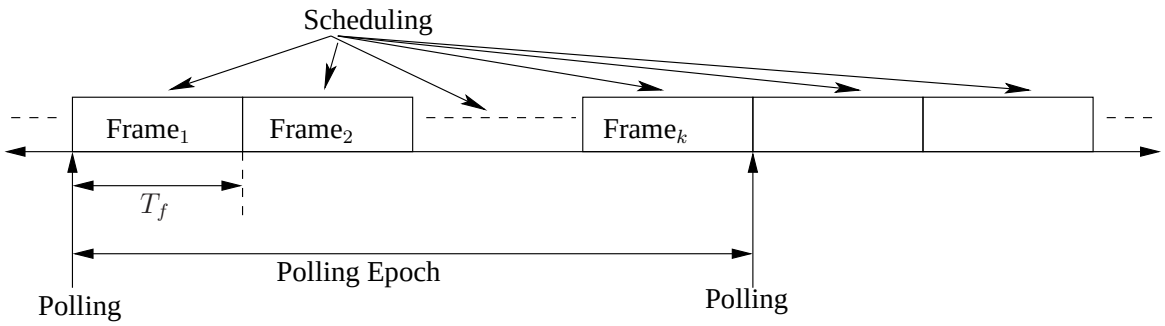


Figure 3.2: Polling and Frame by Frame Scheduling in O-DRR Scheduling

Let $T_{d(i)}$ denote the deadline associated with the packets of SS_i. We define *Normalized*

Deadline as:

$$ND(k) = \frac{T_d}{k \times T_f}, \tag{3.1}$$

where T_d is defined as:

$$T_d = \min_i \{T_{d(i)}\}, \forall i,$$

Note that Normalized Deadline (ND) is a measure of the maximum number of polling epochs that a SS can wait before the expiry of its deadline. If k is chosen such that $k \times T_f > T_{d(i)}$, then delay requirements of SS_i can not be met. However, too small a value of k may incur significant overheads in polling. It is therefore desirable to chose an appropriate value of k.

We assume that the number of slots available for scheduling is greater than the number of connected SS, i.e., $N_s \geq N$. Let i and j be a pair of SSs that are continuously backlogged during an interval (t_1, t_2). Let ϕ_i denote the ideal share of bandwidth to be obtained by SS_i. Slots are assigned proportionately among the SSs in the ratio of their ϕ_is. The *Fairness Measure* $FM(t_1, t_2)$ over all pairs of SSs i and j that are backlogged in the interval (t_1, t_2) is then defined as [29]:

$$FM(t_1, t_2) = \left(\frac{Tx_i(t_1, t_2)}{\phi_i} - \frac{Tx_j(t_1, t_2)}{\phi_j} \right), \tag{3.2}$$

where $Tx_i(t_1, t_2)$ and $Tx_j(t_1, t_2)$ represent the amount of traffic sent in bits by the backlogged flows i and j, respectively, and ϕ_i and ϕ_j represent the bandwidth share of flows i and j, respectively.

If the share of all SSs is equal (when all are backlogged), $\phi_i = \phi_j = 1$, and $\sum_i \phi_i = N$, where N is the total number of connected SSs in the system. Let Q_i be the number of slots that SS_i receives during the time interval (t_1, t_2). Therefore,

$$\phi_i = \frac{Q_i}{Q}, \text{ where } \quad Q = \min_i \{Q_i\} = \frac{T_f}{N}. \tag{3.3}$$

The worst-case occurs when only one SS (say SS_i) is backlogged at the beginning of a polling epoch, and each of the $N - 1$ remaining SS's becomes backlogged *immediately* thereafter. In this case, $Q_i = T_f$ and $\phi_i = N$. If we consider one polling epoch to measure the

fairness, i.e., $t_2 - t_1 = kT_f$, the worst-case fairness measure $|FM_{wc}(t_1, t_2)|$[2] can be expressed as:

$$\begin{aligned} |FM_{wc}(t_1, t_2)| &= \frac{Tx_i(t_2, t_1)}{N} \\ &= \frac{R \times (t_2 - t_1)}{N} \\ &= \frac{R \times k \times T_f}{N}, \end{aligned} \tag{3.4}$$

where R is the maximum data rate in bits/sec achievable by SS_i over the wireless link. From the above equation, we observe that small value of k helps in making the system fair.

3.2.1 Determination of Optimal Polling Epoch k

As explained earlier, the BS needs to poll the SSs to determine the bandwidth and deadline requirements of the SSs after every k frames. We seek to determine an appropriate value for k to minimize a combination of the worst-case relative fairness in bandwidth plus the normalized delay, where the provider may choose the relative weights of the two quantities.

Let $\alpha \times T_f$ be the fraction of a frame of duration T_f that carries uplink transmission where α varies between 0 to 1. Let T_s be the duration of a slot, then $\alpha \times T_f = N_s \times T_s$, where N_s as explained earlier denotes the total number of slots available in a frame for uplink scheduling. We re-write the worst-case fairness from Eqn. (3.4) as:

$$\begin{aligned} |FM_{wc}(t_1, t_2)| &= |FM_{wc}(k \times T_f)| \\ &= \left(\frac{\alpha \times R \times k \times T_f}{N}\right) \\ &= \left(\frac{\alpha \times R \times k \times N_s \times T_s}{N}\right). \end{aligned} \tag{3.5}$$

We would like the optimal k in our solution to be such that the worst-case fairness measure and the normalized delay are minimum. This can be achieved by the following optimization framework[3]:

$$\min_k f(k) = c_1 \times |FM_{wc}(k)| + c_2 \times ND(k), \tag{3.6}$$

[2] $|FM_{wc}(t_1, t_2)|$ denotes the worst-case value of $FM(t_1, t_2)$.

[3] We drop T_f from $|FM_{wc}(k \times T_f)|$ for uniformity.

where c_1 is the cost for a unit of FM per bit and c_2 is the cost per normalized delay. The above equation can be expressed as:

$$\min_k f(k) \quad = a \times k + \frac{b}{k}, \tag{3.7}$$

where $a = c_1 \times \left(\frac{\alpha \times R \times N_s \times T_s}{N}\right)$ and $b = c_2 \times \left(\frac{\alpha \times T_d}{N_s \times T_s}\right)$. The optimal value of k can be obtained at the equilibrium point, where, $c_1 \times |FM'_{wc}(k)| = -c_2 \times ND'(k)$, which simplifies to:

$$k = \sqrt{\frac{b}{a}}, \tag{3.8}$$

If there are flows with different deadlines, then T_ds are different for different classes of traffic. In such cases, the BS can either poll the SSs with different ks, i.e., poll one set of SSs at k_1 another set at k_2 and so on, or poll all SSs with the minimum k. In this chapter, we use the lowest k, i.e., k for the minimum T_d to poll all flows.

With the minimum k, we modify the scheduling algorithm such that the users with loose deadline requirements do not consume resources at the expense of users with tighter deadline requirements. After obtaining the optimum k, we perform uplink bandwidth assignments as explained in the following section.

3.2.2 Slots Assignment

We utilize DRR's idea of maintaining a quantum size Q and a deficit counter DC_i for each SS_i. As discussed before, the quantum size is kept fixed for one polling epoch. The idea of a deficit counter is to ensure fairness among the subscriber stations in the long run. The BS also maintains an indicator variable $Flag_i$ for each SS. $Flag_i$ is 1, if SS_i is assigned slots during a frame, and 0 otherwise. At the beginning of a polling epoch (or at the connection setup), deficit counter of SS_i is initialized to one. Let $N_i(n)$ be the total number of slots assigned to SS_i in frame n of a polling epoch. The deficit counter $DC_i(n)$ is updated as:

$$\begin{aligned} DC_i(0) &= 1 \ \forall i \in L_{sch}, \\ DC_i(n) &= DC_i(n-1) + Q - Flag_i(n-1) \times N_i(n-1), \ \forall i \in L_{sch}, \forall n \geq 1. \end{aligned} \tag{3.9}$$

From the above equation, we observe that the deficit counters of all schedulable SSs are incremented by the quantum Q, whereas the deficit counter of an active SS is decremented by

the amount of slots received by it in the previous frame. Further, we define the scaled deficit counter dc_i for SS_i as follows:

$$\begin{aligned} dc_i(0) =& 1,\ \forall i \in L_{active}, \\ dc_i(n) =& DC_i(n) + \min_j |DC_j(n)|,\ \forall i, j \in L_{active},\ \forall n \geq 1. \end{aligned} \tag{3.10}$$

At the beginning of a polling epoch, the scaled deficit counter is initialized to one.

For each schedulable SS, we define *delay counter* d_i to be a measure of its rate of approach towards the deadline. At the beginning of a polling epoch let $T_{H(i)}$ denote the maximum duration that the Head of the Line (HoL) packet can wait for scheduling before getting dropped due to deadline expiry. Since O-DRR employs no admission control and does not maintain packet level information, strict delay guarantees cannot be provided to each user. Rather, O-DRR only reduces deadline violations than that in RR scheduler by taking the deadlines of HoL packets into account. Note that the maximum value of $T_{H(i)}$ is $T_{d(i)}$, the deadline associated with that packet. In the subsequent frames, the BS updates the delay counters of the schedulable flows as follows:

$$\begin{aligned} d_i(0) =& T_{H(i)},\ \forall i \in L_{sch}, \\ d_i(n) =& d_i(n-1) - T_f,\ \forall i \in (L_{sch} \setminus L_{active}),\ \forall n \geq 1, \\ d_i(n) =& d_i(n-1),\ \forall i \in L_{active},\ \forall n \geq 1. \end{aligned} \tag{3.11}$$

If T_f exceeds $d_i(n)$, then the deadline of the HoL packet that belongs to SS_i has expired. In this scenario, the packet is dropped at the SS and we reset the delay counter value to the maximum permissible delay of SS_i, i.e., $T_{d(i)}$. After computing the scaled deficit counter and the delay counter, the BS determines the weight $W_i(n)$ for SS_i in frame n using:

$$W_i(n) = \frac{\frac{dc_i(n)}{d_i(n)}}{\sum_{j \in L_{active}} \frac{dc_j(n)}{d_j(n)}}, \forall i \in L_{active}. \tag{3.12}$$

For all other SSs, the weight is zero. From Eqn. (3.12), we observe that weight $W_i(n)$ of SS_i is proportional to a normalized product of the deficit counter and the delay counter. This makes intuitive sense, since we would like to give higher bandwidth to a SS that has a smaller delay counter and a higher scaled deficit counter. A small delay counter indicates that a packet in its queue is close to reaching its deadline and a large deficit counter indicates that the SS is

less scheduled as compared to other SSs. After the computation of weights, the number of slots assigned by the BS to SS_i in frame n is determined as:

$$N_i(n) \ = \frac{W_i(n) \times N_s}{\sum_{j \in L_{active}} W_j(n)}, \forall i \in L_{active}. \tag{3.13}$$

In Algorithm 1, we present the pseudo-code of the O-DRR scheduling.

3.3 Implementation of O-DRR Scheduling

In this section, we discuss the implementation of O-DRR in an IEEE 802.16 network. The IEEE 802.16 standard has defined request-grant [3] mechanism in which each SS conveys its bandwidth requirement to the BS. Thus, SSs convey their deadlines at the beginning of a connection to the BS. Note that O-DRR scheduler does not require packet level information while scheduling. SSs are also required to maintain a queue per flow at their interface. If a packet residing in the queue of a SS reaches its deadline, then that packet gets dropped. Packets residing in the queue of a SS are served in a first-come first-serve basis. As per the standard, it is possible for the BS to determine the channel state of each SS. This information is used by the BS to determine the schedulable set at the beginning of polling and to update the active set in every frame.

We consider TDD[4] in which each frame is divided into uplink and downlink subframes of durations T_{ul} and T_{dl} respectively. If $\alpha T_f = T_{ul}$, then $(1-\alpha)T_f = T_{dl}$. In practice, the value of α can be set as 0.5 and T_f can take the value of 0.5 msec, 1 msec or 2 msec for WirelessMAN-SC. The maximum achievable data rate R takes the value of 40 Mbps, 80 Mbps and 120 Mbps with modulation schemes Quadrature Phase Shift Keying (QPSK), 16-Quadrature Amplitude Modulation (16-QAM) and 64-Quadrature Amplitude Modulation (64-QAM) respectively (in WirelessMAN-SC of IEEE 802.16-2004). Using Eqn. (3.6), the optimal value of k can be determined as:

$$\begin{aligned} k &= \min\left(\sqrt{\left(c \times \frac{N \times T_d}{R \times \alpha \times T_f^2}\right)}\right), \\ \text{where } T_d &= \min_i\{T_{d(i)}\}, \forall i, \end{aligned} \tag{3.14}$$

[4]O-DRR can be used for Frequency Division Duplexing (FDD) also.

and c is the ratio of the cost per unit normalized delay to the cost per unit FM, i.e., $c = \frac{c_1}{c_2}$.

Algorithm 1 :O-DRR Scheduling Algorithm with Multi-Class Flows

$\alpha \leftarrow 0.5$
while TRUE **do**
  Determine L_{sch} for the current polling epoch
  $Flag_i(0) \leftarrow 0 \; \forall i \in L_{sch}$
  $DCc_i(0) \leftarrow 1 \; \forall i \in L_{sch}$
  $dc_i(0) \leftarrow 1 \; \forall i \in L_{sch}$
  $d_i(0) \leftarrow T_{H(i)} \; \forall i \in L_{sch}$
  $W_i(0) \leftarrow 0 \; \forall i \in L_{sch}$
  $N_i(0) \leftarrow 0 \; \forall i \in L_{sch}$
  $M \leftarrow |L_{sch}|$
  $Q \leftarrow \frac{N_s}{M}$
  $T_d \leftarrow \min_i\{T_{d(i)}\}, \forall i.$
  $k \leftarrow \min\left(\sqrt{\left(c \times \frac{N \times T_d}{R \times \alpha \times T_f^2}\right)}\right), \forall T_d$
  Frame number $n \leftarrow 1$
  $T \leftarrow k \times T_f$
  while $T > 0$ **do**
    $L_{active} \leftarrow \phi$
    for all $i \in L_{sch}$ **do**
      if $(SINR_i(n) \geq SINR_{th})$ **then**
        $L_{active} \leftarrow L_{active} \cup \{i\}$
        $Flag_i(n) \leftarrow 1$
        $DC_i(n) \leftarrow DC_i(n-1) + Q - Flag_i(n-1) \times N_i(n-1)$
      else
        $Flag_i(n) \leftarrow 0$
        $DC_i(n) \leftarrow DC_i(n-1) + Q$
        $W_i(n) \leftarrow 0$
        $N_i(n) \leftarrow 0$
        $d_i(n) \leftarrow d_i(n-1) - T_f$
        if $d_i(n) \leq 0$ **then**
          $d_i(n) \leftarrow T_{d(i)}$
        end if
      end if
    end for
    for all $i \in L_{active}$ **do**
      $dc_i(n) \leftarrow DC_i(n) + \min_j |DC_j(n)|, \forall j \in L_{active}$
      $W_i(n) \leftarrow \frac{\frac{dc_i}{d_i}}{\sum_{j \in L_{active}} \frac{dc_j}{d_j}}$
      $N_i(n) \leftarrow \frac{W_i(n) \times N_s}{\sum_{j \in L_{active}} W_j(n)}$
      $d_i(n) \leftarrow d_i(n-1)$
    end for
    $T \leftarrow T - T_f$
    $n \leftarrow n + 1$
  end while
end while

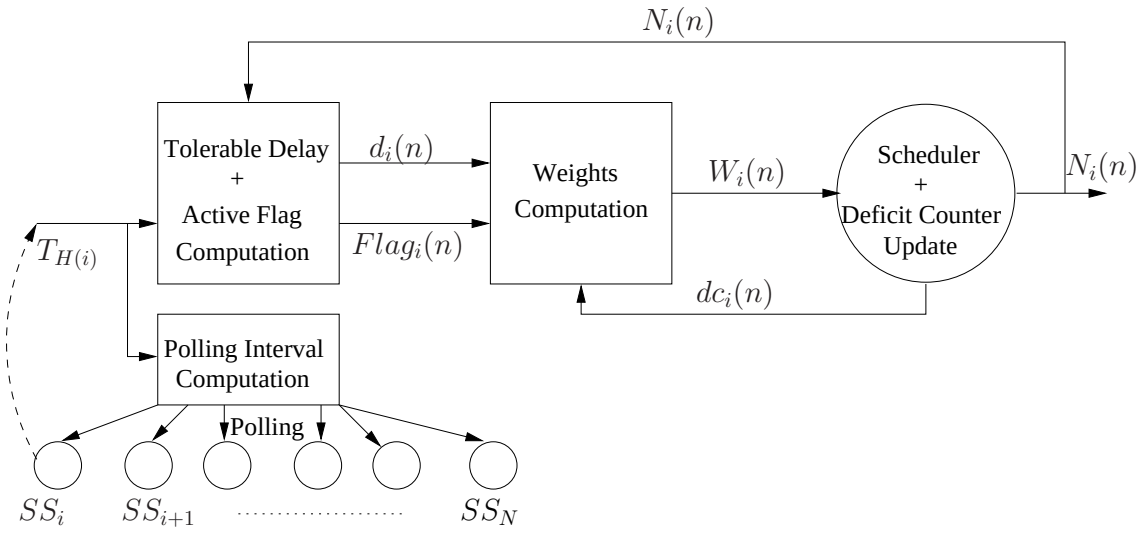


Figure 3.3: Block Diagram of the O-DRR Scheduler

At every polling epoch, each SS is required to communicate its deadline associated with the HoL packet ($T_{H(i)}$) with the BS. For scheduling, the BS maintains one deficit counter, one scaled deficit counter and one delay counter for each connected SS. It determines the weights and assign slots to the active set members in every frame. This slot assignment information or grant is conveyed to SSs through the UL_{MAP} in every frame. In Figure 3.3, we explain the block diagram of this cross-layer implementation.

3.3.1 An Example of O-DRR Scheduling Scheme

In this section, we illustrate O-DRR scheduling through an example. We consider six SSs and a BS in a multipoint-to-point scenario. Let the frame duration T_f be 5 msec and the polling epoch k be 3 frames. Let the deadline $T_{d(i)}$ associated with the packets be 30 msec. Let the SNR_{th} be 25 dB and the total number of schedulable slots (N_s) in a frame be 60.

We assume that at the first polling epoch, all SSs have packets to transmit. At the beginning of a polling epoch, let the SNR of the six SSs be 31, 30, 28, 35, 26 and 32 dB respectively. Since the SNR of all SSs are above SNR_{th} and the SSs have packets to transmit, all of them form the schedulable set. Hence, at the beginning of first polling epoch $M = 6$. The quantum Q for the first polling epoch is 10. Let at the time of polling, the deadlines ($T_{H(i)}$) associated with the HoL packets of six SSs be 10, 30, 20, 25, 18 and 20 msec respectively. At the beginning of the polling, the deficit counter DC_i and scaled deficit counter dc_i for each member of

the schedulable set are set to one. In the first frame, all six SSs form the active set (same as schedulable set, Figure 3.4). BS then determines the weight of each schedulable SS using Eqn. (3.12) and assigns slots to the active SSs using Eqn. (3.13). The number of slots assigned to each SS is given in Table 3.1.

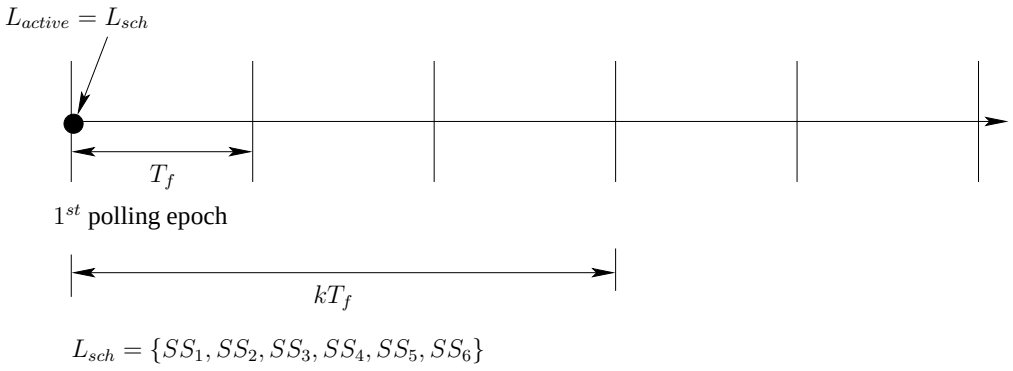


Figure 3.4: Polling and Scheduling, 1^{st} frame of 1^{st} Polling Epoch

Table 3.1: Operation of O-DRR Scheduler, 1^{st} frame of 1^{st} Polling Epoch

SS_i	SNR_i	L_{sch}	L_{active}	$DC_i(0)$	$dc_i(0)$	$d_i(0)$	$N_i(1)$
1	31	1	1	1	1	10	18
2	30	1	1	1	1	30	6
3	28	1	1	1	1	20	9
4	35	1	1	1	1	25	8
5	26	1	1	1	1	18	10
6	32	1	1	1	1	20	9

At the beginning of the second frame, let the SNR of six SSs be 31, 30, 20, 35, 23 and 32 dB respectively. Since the SNR of SS_3 and SS_5 are less than SNR_{th}, they are excluded from the active set. The other four SSs now constitute the active set (see Figure 3.5). BS updates the deficit counters and delay counters of each schedulable set member, scaled deficit counter of each active set member and assigns slots to the active set members as given in Table 3.2.

From Table 3.2, we observe that the delay counters of SS_3 and SS_5 are decremented by a frame duration ($\in L_{sch} \setminus L_{active}$), whereas the delay counters of other SSs remains unchanged. At the beginning of the third frame, let the SNR of six SSs be 24, 27, 26, 30, 26 and 30 dB respectively. Since the SNR of SS_1 is less than SNR_{th}, SS_1 is excluded from the active set.

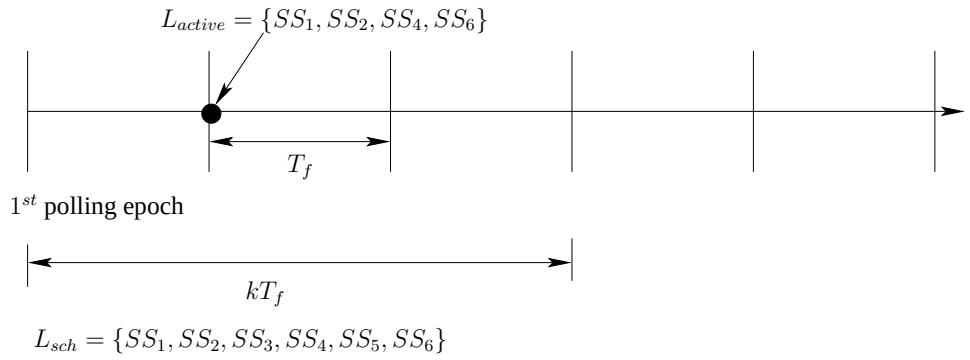


Figure 3.5: Polling and Scheduling, 2^{nd} frame of 1^{st} Polling Epoch

Table 3.2: Operation of O-DRR Scheduler, 2^{nd} frame of 1^{st} Polling Epoch

SS_i	SNR_i	L_{sch}	L_{active}	$DC_i(1)$	$dc_i(1)$	$d_i(1)$	$N_i(2)$
1	31	1	1	-7	0	10	0
2	30	1	1	5	12	30	20
3	20	1	0	2	NA	20	0
4	35	1	1	3	10	25	19
5	23	1	0	1	NA	18	0
6	32	1	1	2	9	20	21

The other five SSs now constitute the active set. The BS then determines the deficit counter and delay counter for each schedulable SS, scaled deficit counter for each active SS. The BS then determines the weights of each schedulable SS using Eqn. (3.12) and assigns slots to the active SSs using Eqn. (3.13).

From Table 3.3, we observe that the delay counter of SS_1 is decremented by a frame duration, whereas the delay counters of all other SSs remain constant. Since we have chosen $k = 3$, the BS polls again after the end of the third frame (second polling epoch) to gather the deadline information of the HoL packet of the connected set. Let the SNR of six SSs be 30, 24, 26, 28, 31 and 23 dB respectively. Since the SNR of SS_2 and SS_6 are less than SNR_{th}, they will not be polled successfully. Instead, SS_1, SS_3, SS_4 and SS_5 are polled successfully and constitute the schedulable set for the second polling epoch (Figure 3.7). In this polling epoch $M = 4$, therefore the quantum size $Q = 15$. At the beginning of the second polling epoch, the BS resets the deficit counters and the scaled deficit counters of the members of the schedulable set to one. Let at the beginning of second polling epoch, the deadlines associated with the HoL

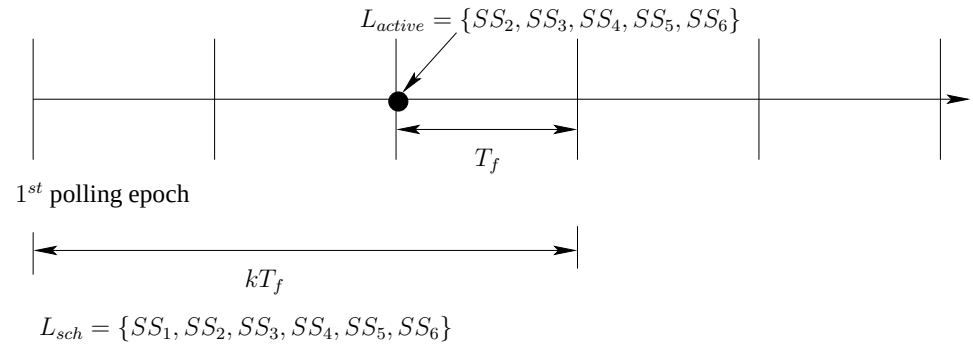


Figure 3.6: Polling and Scheduling, 3^{rd} frame of 1^{st} Polling Epoch

Table 3.3: Operation of O-DRR Scheduler, 3^{rd} frame of 1^{st} Polling Epoch

SS_i	SNR_i	L_{sch}	L_{active}	$DC_i(2)$	$dc_i(2)$	$d_i(2)$	$N_i(3)$
1	24	1	0	3	NA	10	0
2	27	1	1	-5	4	30	3
3	26	1	1	12	21	15	26
4	30	1	1	-6	3	25	2
5	26	1	1	11	20	13	29
6	30	1	1	-9	0	20	0

packets of six SSs be 5, 30, 20, 25, 15 and 20 msec respectively. Note that SS_2 and SS_6 will not be considered for scheduling in the second poling epoch. In this example, since $T_{H(2)}$ and $T_{H(6)}$ are greater than one polling epoch ($kT_f = 15$ msec), HoL packets of SS_2 and SS_6 will not be dropped before the next polling epoch.

The BS then determines the weights of each schedulable SS using Eqn. (3.12) and assigns slots to the active SSs using Eqn. (3.13). It updates the deficit counters and the delay counters for each schedulable SS and the scaled deficit counter for each active SS in the next frame. This process continues in every frame of each polling epochs.

3.4 Experimental Evaluation of O-DRR Scheduling

In this section, we describe simulation experiments that have been performed to evaluate O-DRR scheduling. All the simulations have been conducted using implementations of O-DRR scheduling in IEEE 802.16 setting in MATLAB [68]. We consider a multipoint-to-point IEEE 802.16 network where 100 SSs are connected to a centralized BS as shown in Figure 3.1. We further consider WirelessMAN-SC air interface as an example, and use QPSK modulation

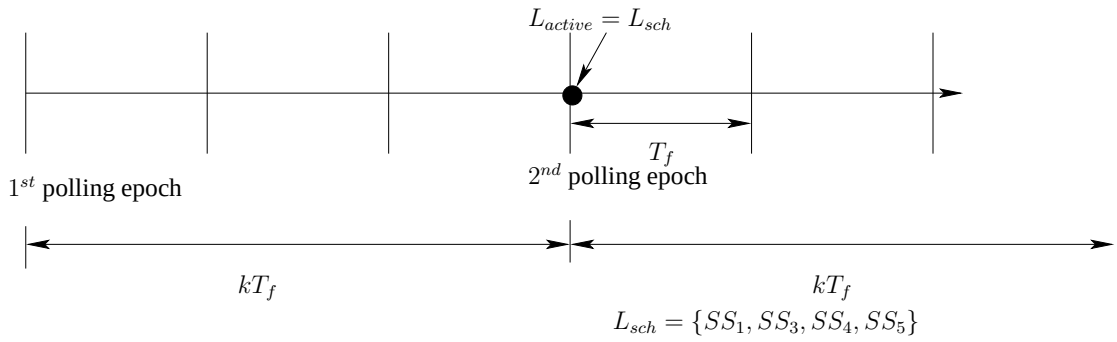


Figure 3.7: Polling and Scheduling, 1^{st} frame of 2^{nd} Polling Epoch

Table 3.4: Operation of O-DRR Scheduler, 1^{st} frame of 2^{nd} Polling Epoch

SS_i	SNR_i	L_{sch}	L_{active}	$DC_i(0)$	$dc_i(0)$	$d_i(0)$	$N_i(1)$
1	30	1	1	1	1	5	34
2	24	0	0	NA	NA	NA	NA
3	26	1	1	1	1	20	8
4	28	1	1	1	1	25	7
5	31	1	1	1	1	15	11
6	23	0	0	NA	NA	NA	NA

scheme (mandatory as per the standard) between SSs and the BS. We consider two different sets of experiments. In the first experiment, the deadlines T_ds of all flows are the same with T_d = 200 msec. We term this experiment as "single-class" (SC) experiment. In the second experiment, SSs are divided evenly into two-classes based on their deadline requirements, $T_{d(1)}$ = 200 msec and $T_{d(2)}$ = 500 msec. We term this experiment as "multi-class" (MC) experiment. Each SS is assumed to have a large enough buffer such that packets get dropped at a SS only due to deadline violation. The frame duration T_f is set to 1 msec, with N_s = 100 slots.

The path loss exponent due to distance is set as $\gamma = 4$. We simulate both shadowing as well as fast fading in our experiments. We also consider Additive White Gaussian (AWGN) with Power Spectral Density (PSD) N_0 = 0.35 (4.5 dB/Hz). The shadowing is modeled as Log-normal with mean zero and standard deviation (σ) of 8 dB. In each simulation run, the channel gain due to Log-normal shadowing is kept fixed for a duration of 50 frames. For fast fading, we consider Rayleigh fading model. The channel gain due to fast fading is modeled as complex Gaussian random variable or equivalently the power gain is an exponential random variable

with mean β. The coherence time of Rayleigh fading is considered to be equal to one frame duration, i.e, the channel gain due to fast fading changes from frame to frame. The value of β and transmission power is chosen such that the expected SNR received[5] due to Log-normal shadowing, Rayleigh fading and path loss for a SS at the cell edge is more than SNR_{th} required for transmission. We consider $SNR_{th} = 12.18$ dB, the minimum SNR required for IEEE 802.16-2004 WirelessMAN-SC air interface in our simulations. We also repeat the experiments with different Log-normal shadowing with σ of 4, 6, 8, 10 and 12 dB.

We model both video traffic [69] and general web traffic [50, 70]. To generate video traffic [69] we use self-similar traffic model based on the random midpoint displacement algorithm [71]. The self-similar video traffic is generated as fractional Brownian noise (fBn) with Hurst parameter $H = 0.8$, mean rate to peak rate ratio $\rho = 0.3$ around the unity link capacity. We consider fixed packet lengths of 100 bits.

For web-traffic, we generate variable sized packets drawn from a truncated Pareto distribution [50, 70]. This distribution is characterized by three parameters: shape factor ξ, mode υ and cutoff threshold φ_{th}. The probability that a packet has a size ℓ can be expressed as:

$$\begin{aligned} f_{TP}(\ell) &= \frac{\xi \cdot \upsilon^{\xi}}{\ell^{\xi+1}}, \quad \upsilon \leq \ell < \varphi_{th} \\ f_{TP}(\ell) &= \eta, \qquad\qquad \ell \geq \varphi_{th}, \end{aligned} \tag{3.15}$$

where η can be calculated as:

$$\eta = \left(\frac{\xi}{varphi_{th}} \right)^{\xi}, \ \xi > 1. \tag{3.16}$$

We choose shape factor $\xi = 1.2$, mode $\upsilon = 50$ bits, cutoff threshold $\varphi_{th} = 500$ bits which provides us an average packet size of 110 bits. In each frame, we generate the arrivals for all the users using Poisson distribution. Arrivals are generated in an independent and identically distributed (i.i.d.) manner across the frames. We also use a mix traffic case in which approximately 50% of the SSs generate Video traffic and the remaining 50% generate Pareto traffic. The mean rate of the traffic source is scaled relative to the achievable data rate R to achieve effective link utilization. We term this as the load of the system.

We determine the optimal polling epoch for both "single-class" and "multi-class" experiments. We use the smaller k which corresponds to $T_{d(1)}$ in both single-class and multi-class

[5]The expected signal power received at a distance d can be determined as: $P_r = P_t d^{-\gamma} \beta e^{(\frac{(\log 10\sigma)^2}{200})}$. For details refer to Section 2.1.

experiments. With R = 40 Mbps, N = 100, α = 0.5, T_d = 200 msec and T_f = 1 msec, we determine different values of k with different cost c. We evaluate the effect of choosing different polling epoch on packet drops and fairness.

Each simulation has been performed for a heavy network load to stress test the performance of the O-DRR scheduler over a period of 2500 frames (250,000 slots). We also vary the load from 70% to 95% to evaluate the performance of O-DRR scheduler at different load conditions. The value of each parameter observed has been averaged over 50 independent simulation runs, with the "warm up" frames (approximately 500 frames) being discarded in each run, to ensure that the values observed are steady-state values. The simulation parameters are provided in Table 3.5.

Table 3.5: Summary of Simulation Parameters in O-DRR Scheduling

Simulation Parameter	**Value**
Modulation Scheme	QPSK
Frame Duration (T_f)	1 msec
Number of SSs	100
Number of Frames	2500
Number of Uplink Slots per Frame (N_s)	100
H	0.8
ρ	0.3
ξ	1.2
υ	50 bits
φ_{th}	500 bits
$T_{d(1)}$	200 msec
$T_{d(2)}$	500 msec
Load	70, 75, 80, 85, 90, 95%
σ (Log-normal shadowing)	4, 6, 8, 10, 12 dB
Poling Interval (k)	10-120

3.4.1 Simulation Results

In this section, we analyze the performance of O-DRR scheduler in terms of percentage of packets dropped and fairness. To measure fairness, we use Jain's Fairness Index [65].

Effect of Polling Epoch k on Packet Drops

We simulate with k varying from 10 to 120 to assess the performance of O-DRR scheduler at different polling epochs for both Video and Pareto traffic. In Figures 3.8 - 3.9, we plot the percentage of packet drops with varying polling epoch k for different values of load. From these figures, we observe a concave relationship between percentage of packets dropped and k. The high percentage of packets dropped at small k is due to the large overheads of polling while at high k, the packet drop is due to the deadline expiry. From these figures, we observe that at a relatively higher load (90, 95 and 99%) the percentage of packets dropped reaches its minimum at $k = 50$ for both Video and Pareto traffic. Therefore, we use $k = 50$ for the rest of the simulations.

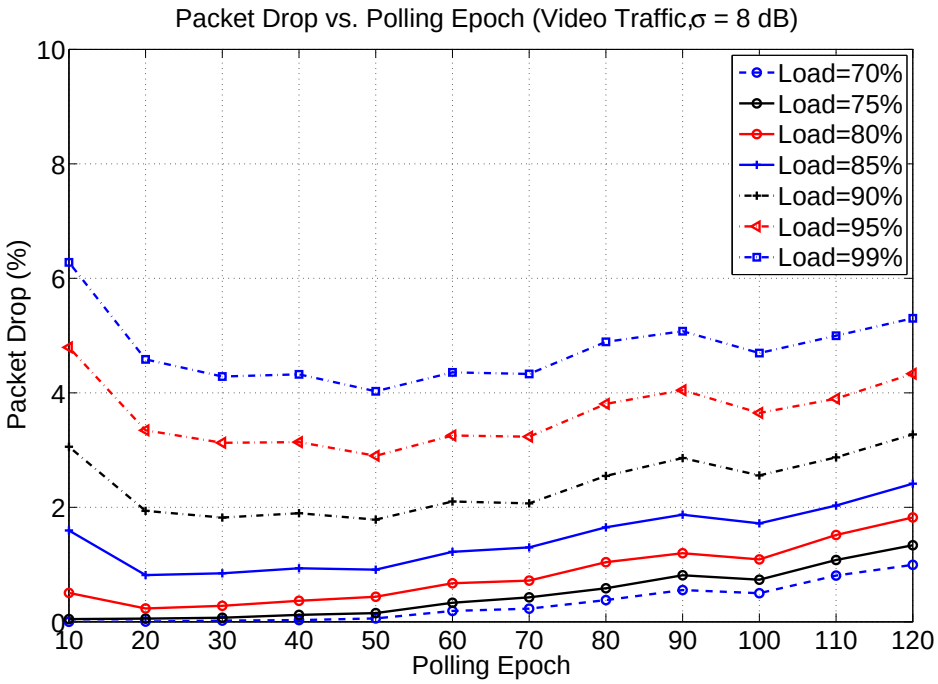


Figure 3.8: Percentage of Packets Dropped at Different Polling Epochs (Video)

Fairness of O-DRR Scheduler

To assess the fairness of the O-DRR algorithm, we determine the Jain's Fairness Index (JFI) for both "single-class" and "multi-class" experiments. In Figures 3.10 and 3.11, we plot JFI of O-DRR scheduler at different loads, keeping the Log-normal shadowing parameter constant ($\sigma = 8$ dB). From these figures, we observe that JFI is above 98% for all traffic cases (Video, Pareto and Mixed). We also observe that JFI decreases with increase in load.

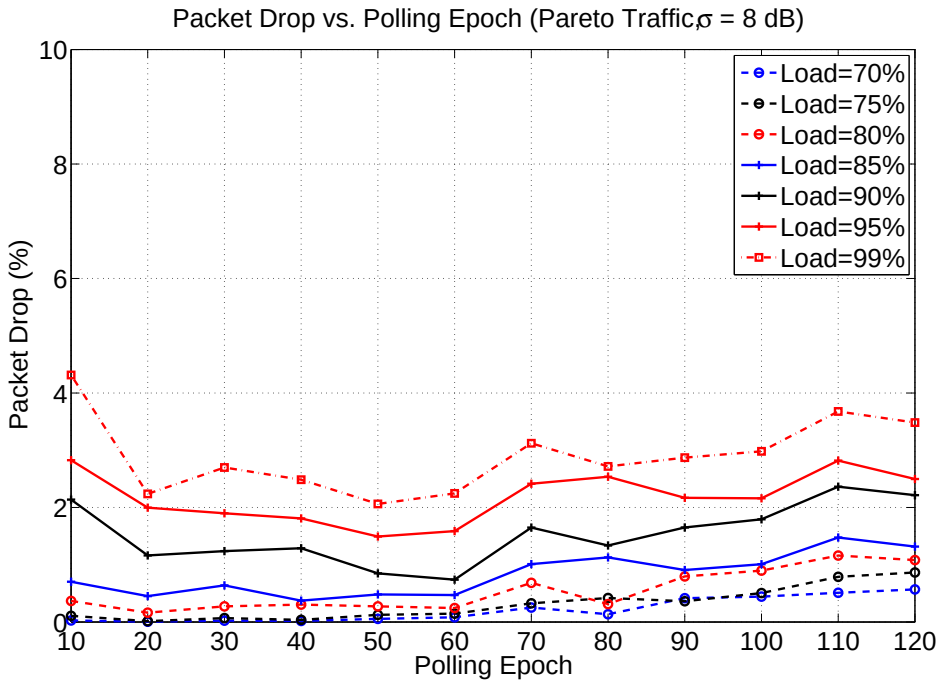


Figure 3.9: Percentage of Packets Dropped at Different Polling Epochs (Pareto)

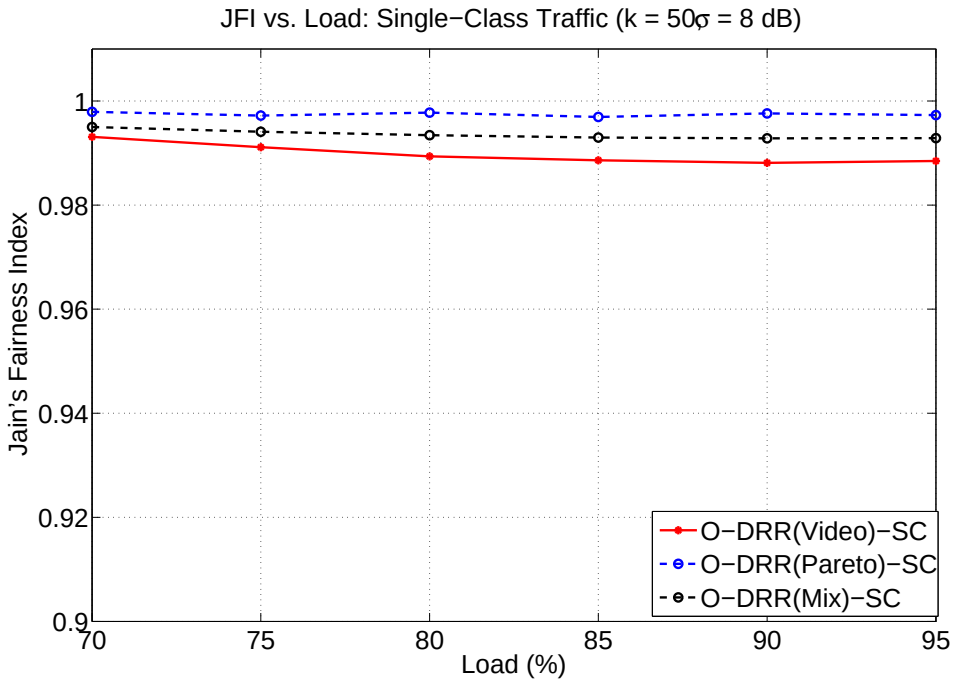


Figure 3.10: Jain's Fairness Index at Different Loads: Single-Class Traffic

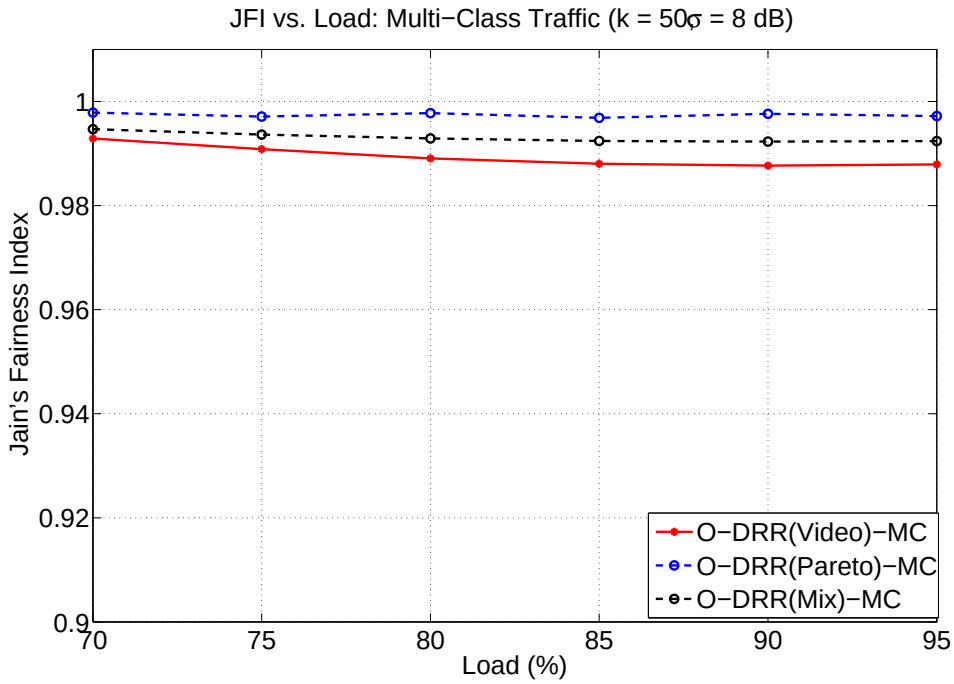


Figure 3.11: Jain's Fairness Index at Different Loads: Multi-Class Traffic

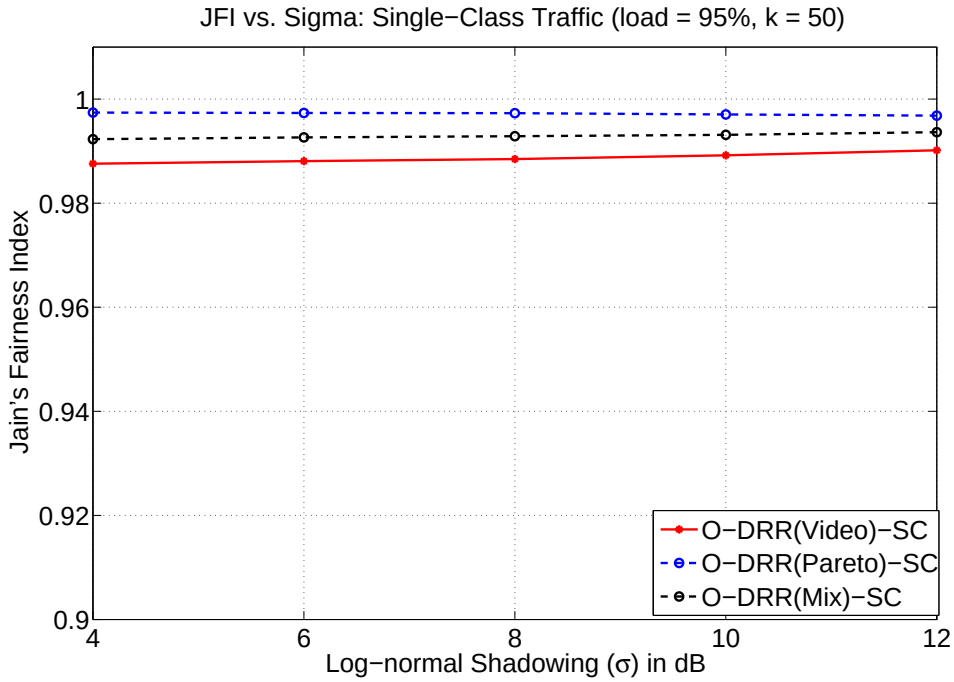


Figure 3.12: Jain's Fairness Index at Different Log-normal Shadowing: Single-Class Traffic

Figure 3.13: Jain's Fairness Index at Different Log-normal Shadowing: Multi-Class Traffic

In Figures 3.12 and 3.13, we plot JFI of O-DRR scheduler for different values of Log-normal shadowing, $\sigma = 4, 6, 8, 10$ and 12 dB, with the load being fixed at 95%. From these figures, we observe that JFI is above 98% for all traffic cases. From Figures 3.10 - 3.13, we observe that JFI is above 98% even at 95% load for all values of σ.

3.4.2 Comparison with Round Robin Scheduler

We compare the performance of O-DRR scheduler with a simple Round Robin (RR) scheduler. Like the O-DRR scheduler, we also use the polling method to determine the schedulable list at the beginning of the polling epoch for the RR scheduler. The polling epoch k used in RR scheduler is same as that used in O-DRR scheduler. Unlike O-DRR scheduler, however, in RR scheduler, the BS schedules the active SSs in every frame in a simple round robin manner. In this section, we compare the percentage of packets dropped and JFI of O-DRR scheduler with that of RR scheduler at: (a) Different Loads; (b) Different Log-normal Shadowing; and (c) Different Polling Epochs.

For Varying Values of Load

In Figures 3.14, 3.15 and 3.16, we plot the percentage of packets dropped observed in both RR and O-DRR scheduler at different loads for Video, Pareto and Mixed traffic respectively. The

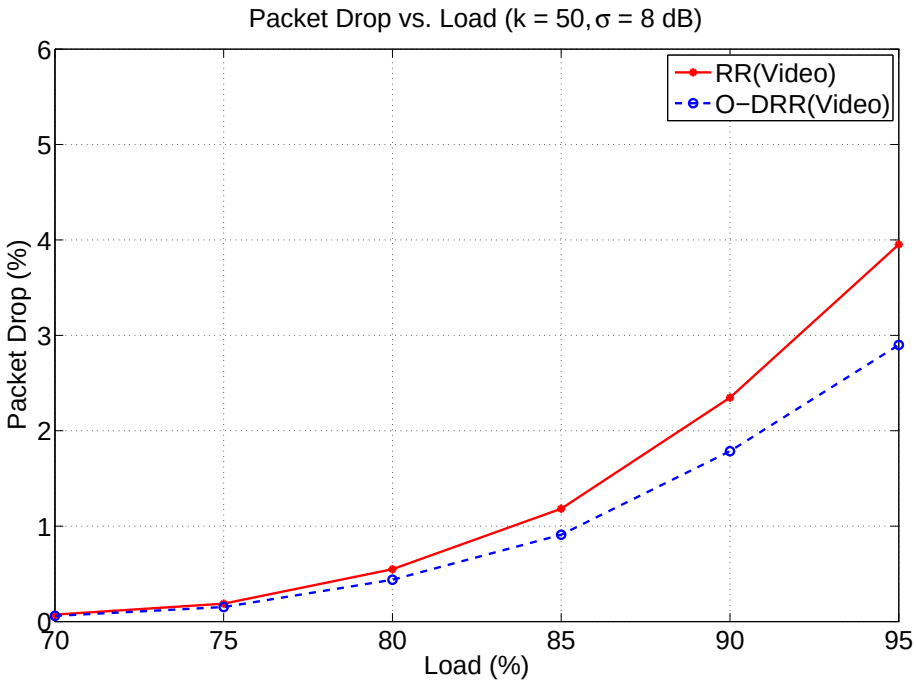


Figure 3.14: Percentage of Packets Dropped at Different Loads with Multi-Class Video Traffic

value of k and σ are kept constant ($k = 50$ and $\sigma = 8$ dB) in these figures. From these figures, we observe that the O-DRR scheduler out-performs RR scheduler in packet drop percentage for all traffic cases. Since O-DRR is a deadline based scheduler, flows with relatively smaller deadlines are scheduled with higher priority than the flows with higher deadlines. This improves the performance of the scheduler and minimizes packet drops due to deadline violation, which is reflected in these figures. In Figure 3.17, we plot the percentage gain (improvement) in packets dropped of O-DRR scheduler over that of RR scheduler at different loads. From this figure, we observe that the O-DRR scheduler performs better at relatively high loads as compared to relatively low loads. We also observe that when the load is 95%, the improvements of O-DRR scheduler over RR scheduler are around 37%, 27% and 18% for Video, Mixed and Pareto traffic respectively. The high relative improvement of packet drop percentage of O-DRR scheduler over RR scheduler signifies the superiority of O-DRR scheduler over RR scheduler in an IEEE 802.16 network.

Moreover, from Figures 3.18, 3.19 and 3.20, we observe that JFI achieved by O-DRR scheduler is more than that of RR scheduler. This signifies the fairness properties of O-DRR scheduler as compared to RR scheduler at different loads.

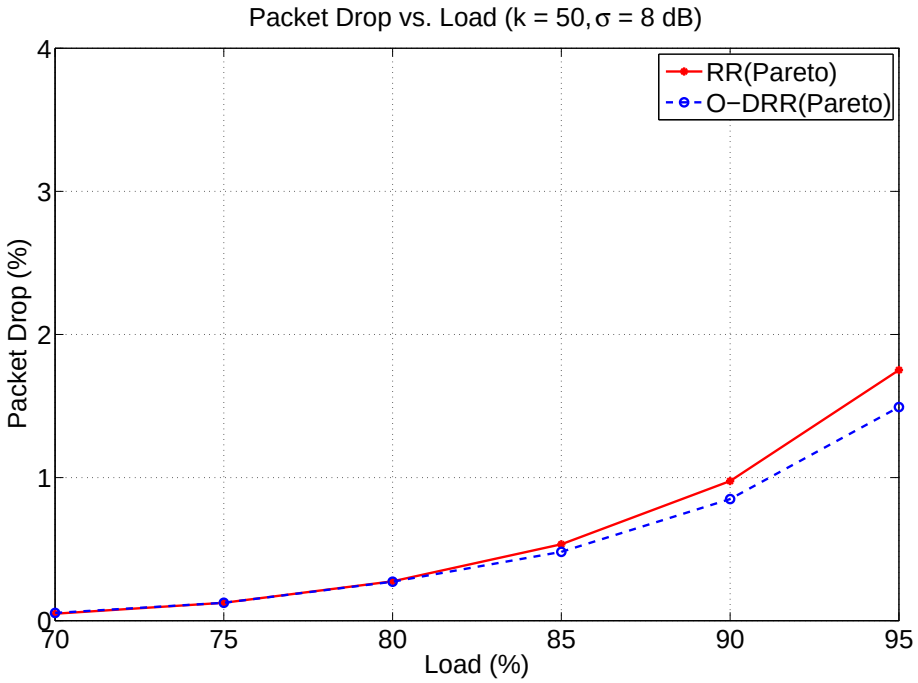


Figure 3.15: Percentage of Packets Dropped at Different Loads with Multi-Class Pareto Traffic

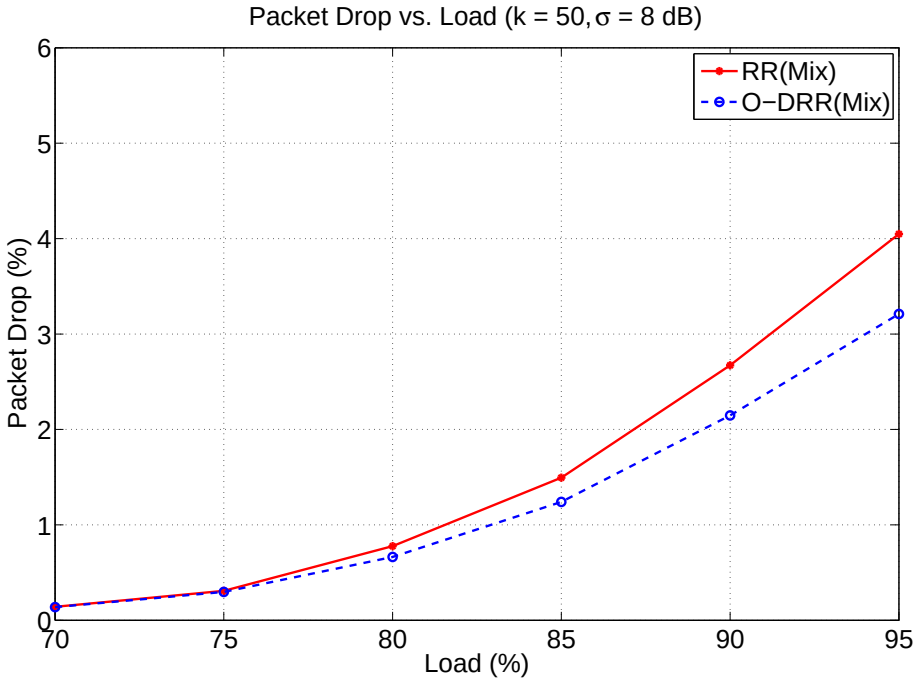


Figure 3.16: Percentage of Packets Dropped at Different Loads with Multi-Class Mixed Traffic

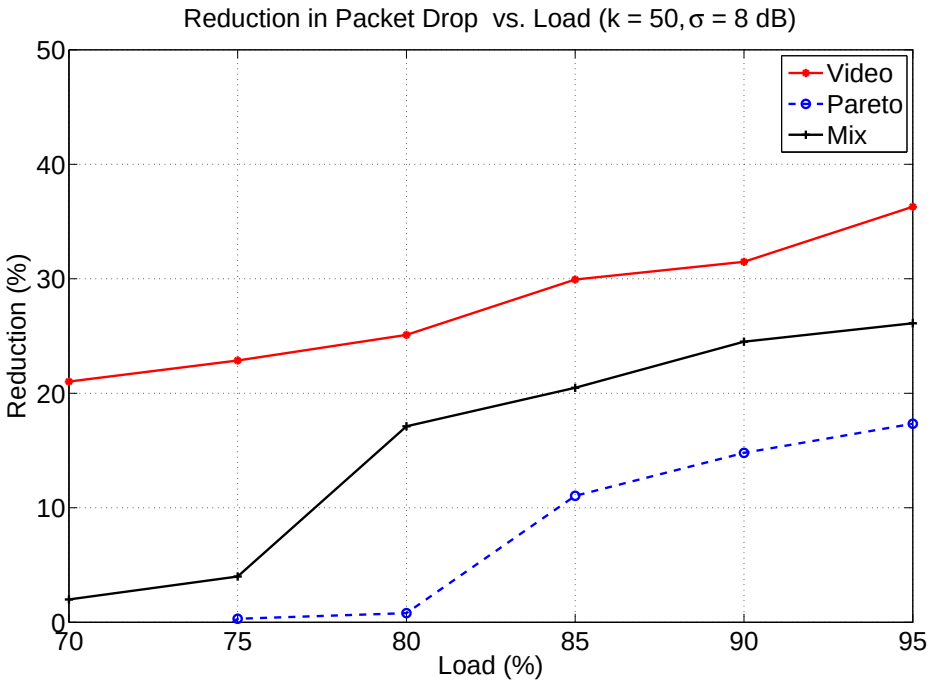


Figure 3.17: Performance Gain of O-DRR Scheduler over RR Scheduler in terms of Packet Drops at Different Loads

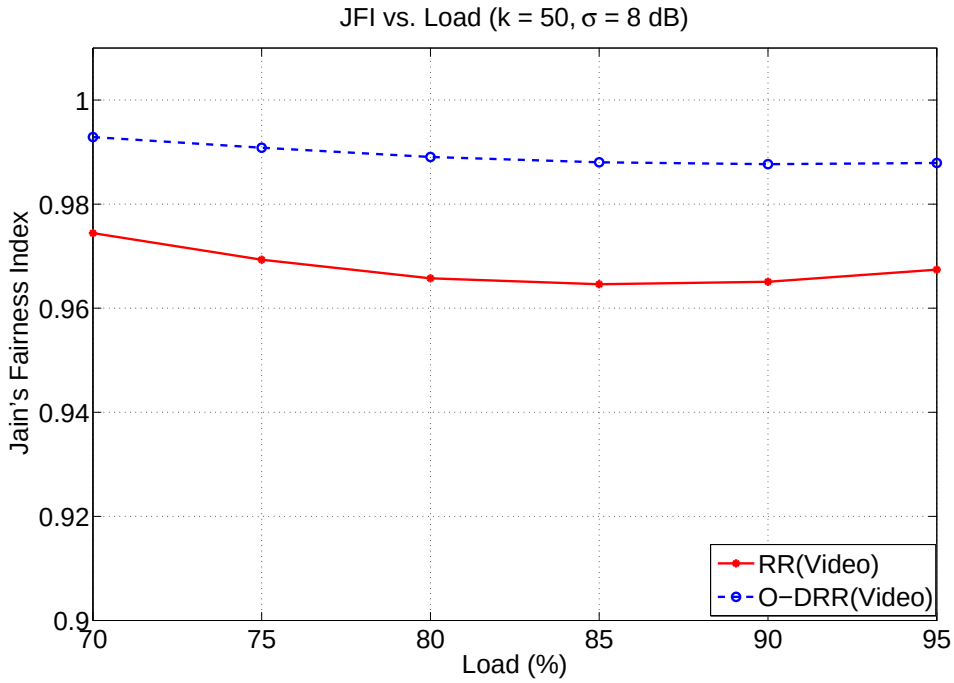


Figure 3.18: Jain's Fairness Index at Different Loads with Multi-Class Video Traffic

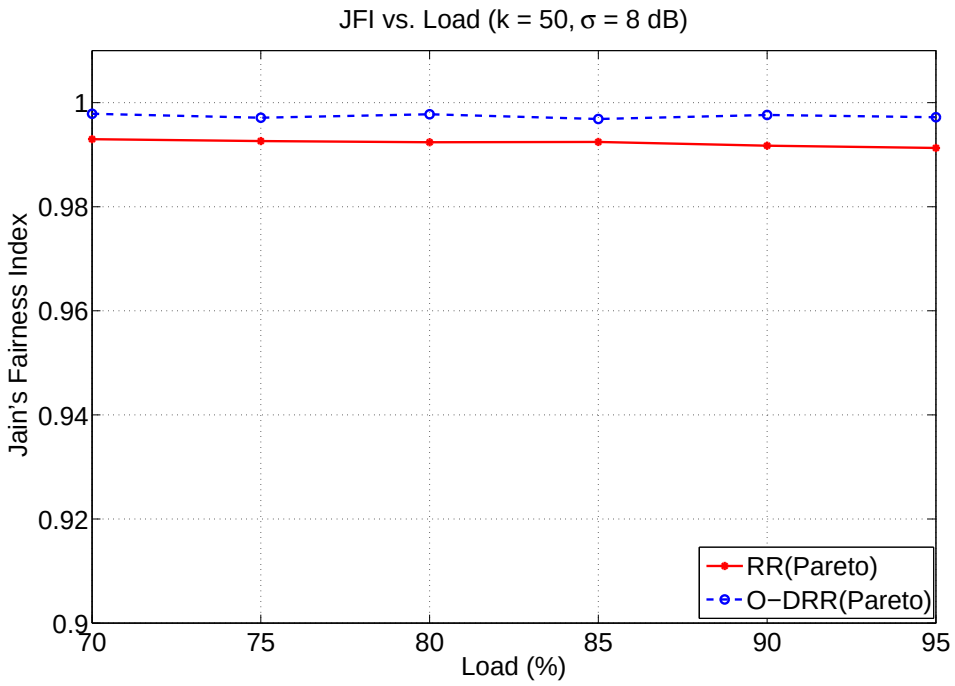


Figure 3.19: Jain's Fairness Index at Different Loads with Multi-Class Pareto Traffic

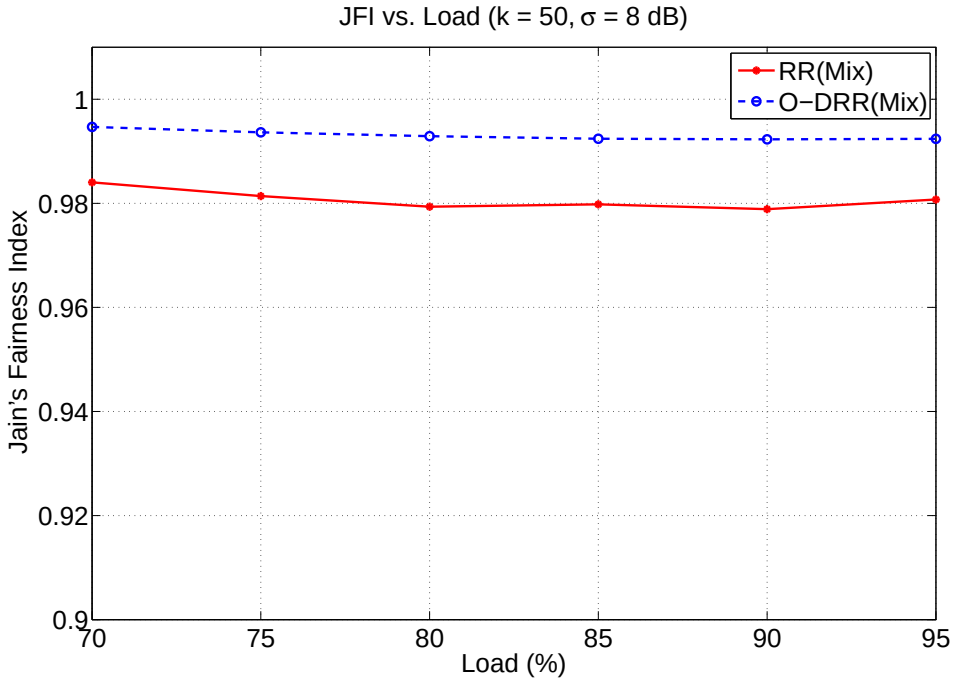


Figure 3.20: Jain's Fairness Index at Different Loads with Multi-Class Mixed Traffic

For Varying Values of Standard Deviation (σ) of Log-Normal Shadowing

We also vary the σ of Log-normal shadowing from 4 dB to 12 dB to determine the robustness of the O-DRR scheduler compared to that of RR scheduler. In Figures 3.21 and 3.22, we plot the percentage of packets dropped observed for Video and Pareto traffic respectively in both RR and O-DRR scheduler at different values of Log-normal shadowing. The values of k and load are kept constant ($k = 50$, load = 95%) in these figures. From these figures, we observe that the O-DRR scheduler out-performs RR scheduler in packet drop percentage for both traffic cases. In Figure 3.23, we plot the percentage gain (improvement) in packets dropped of O-DRR scheduler over RR scheduler for different values of σ of Log-normal shadowing. From this figure, we observe that the O-DRR scheduler performs better than the RR scheduler even when the Log-normal shadowing is high. We also observe that at 95% load and at $\sigma = 12$ dB, the improvements of O-DRR scheduler over RR scheduler are more than 40% for both Video and Pareto traffic. Therefore, O-DRR scheduler is more suitable for all load, all fading and traffic requirements.

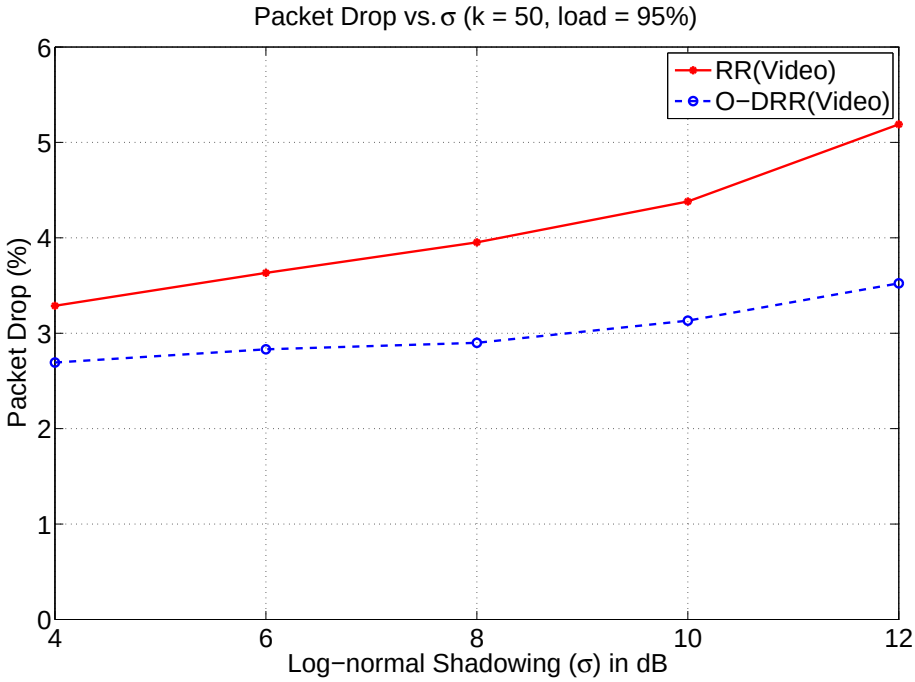


Figure 3.21: Percentage of Packets Dropped at Different Log-normal Shadowing with Multi-Class Video Traffic

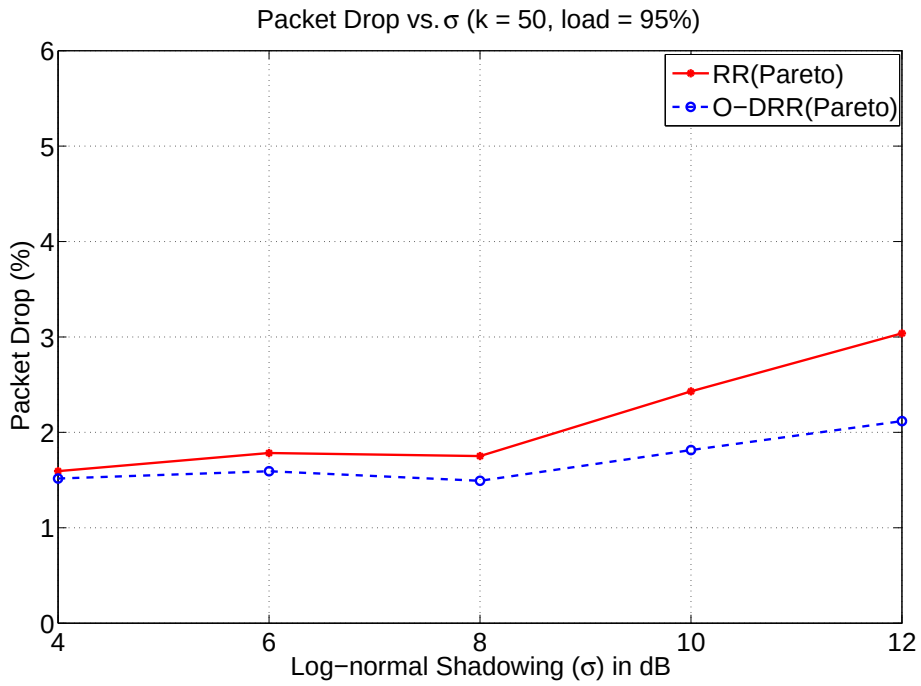


Figure 3.22: Percentage of Packets Dropped at Different Log-normal Shadowing with Multi-Class Pareto Traffic

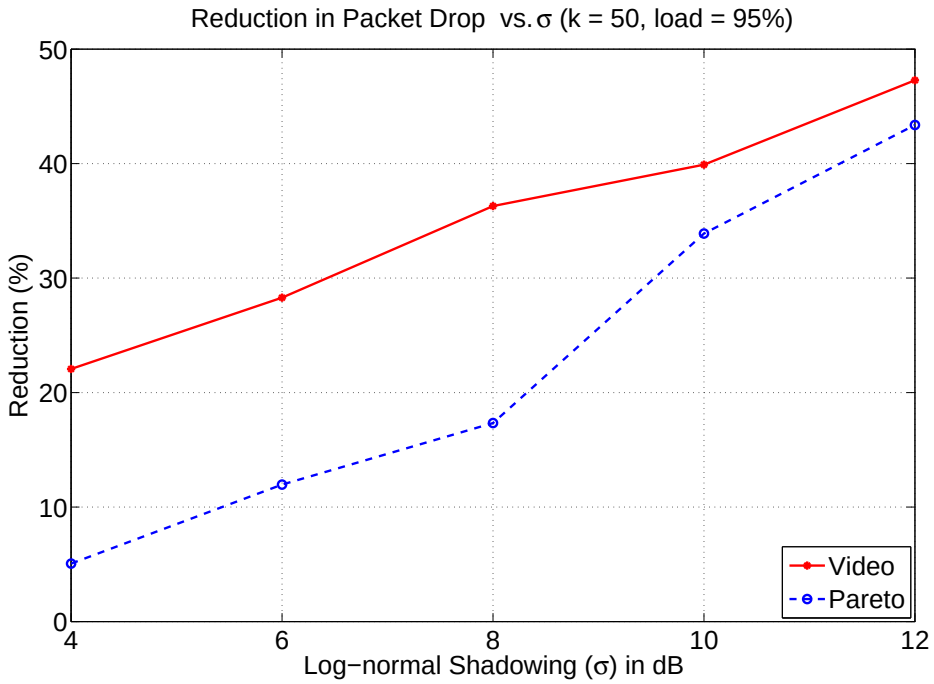


Figure 3.23: Performance Gain of O-DRR Scheduler over RR Scheduler in terms of Packet Drops at Different Log-normal Shadowing

Moreover, from Figures 3.24 and 3.25, we observe that JFI achieved by O-DRR scheduler is more than that of RR scheduler. This signifies the fairness properties of O-DRR scheduler as compared to that of RR scheduler for different values of σ of Log-normal shadowing.

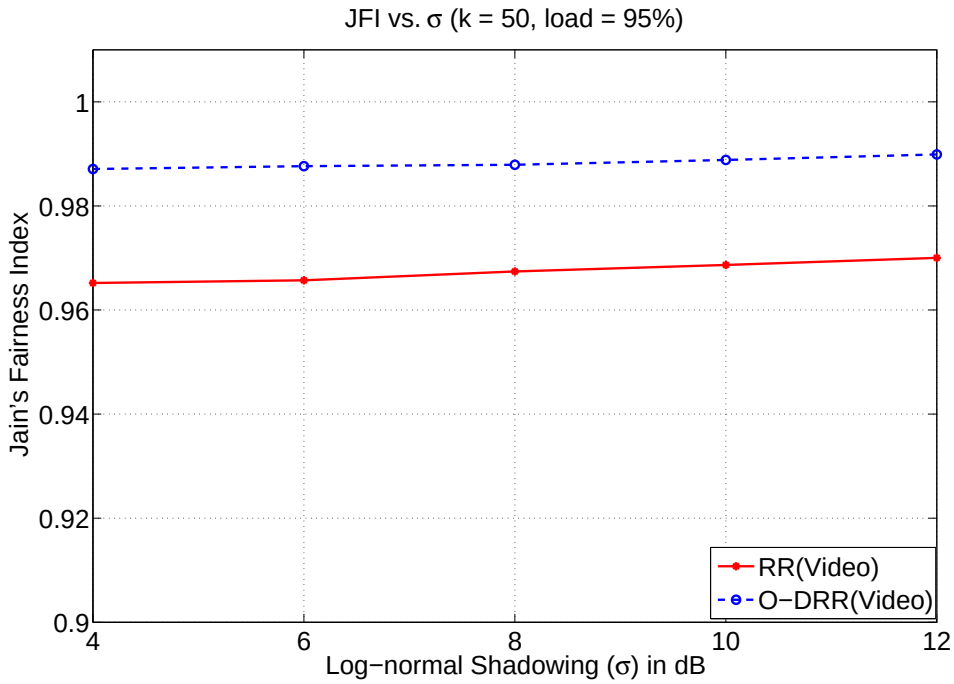


Figure 3.24: Jain's Fairness Index at Different Log-normal Shadowing with Multi-Class Video Traffic

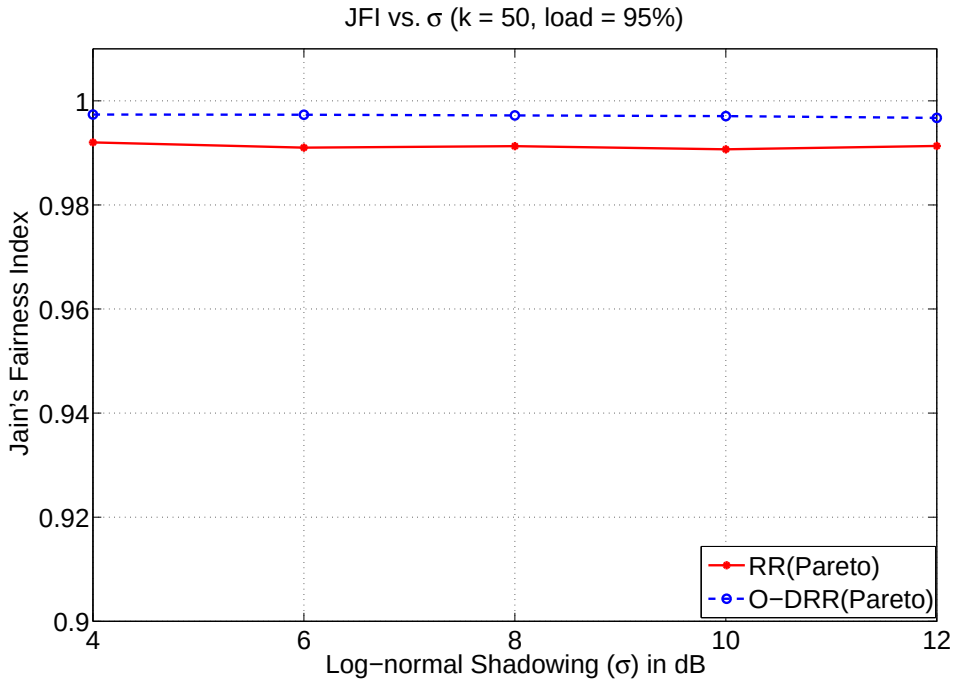


Figure 3.25: Jain's Fairness Index at Different Log-normal Shadowing with Multi-Class Pareto Traffic

For Varying Values of Polling Epoch k

We also compare JFI achieved by O-DRR with that of RR at different values of k, keeping $\sigma = 8$ dB and load = 95% as fixed. The results are plotted in Figures 3.26 and 3.27. We observe that JFI achieved by O-DRR is more than that of RR at different polling epochs.

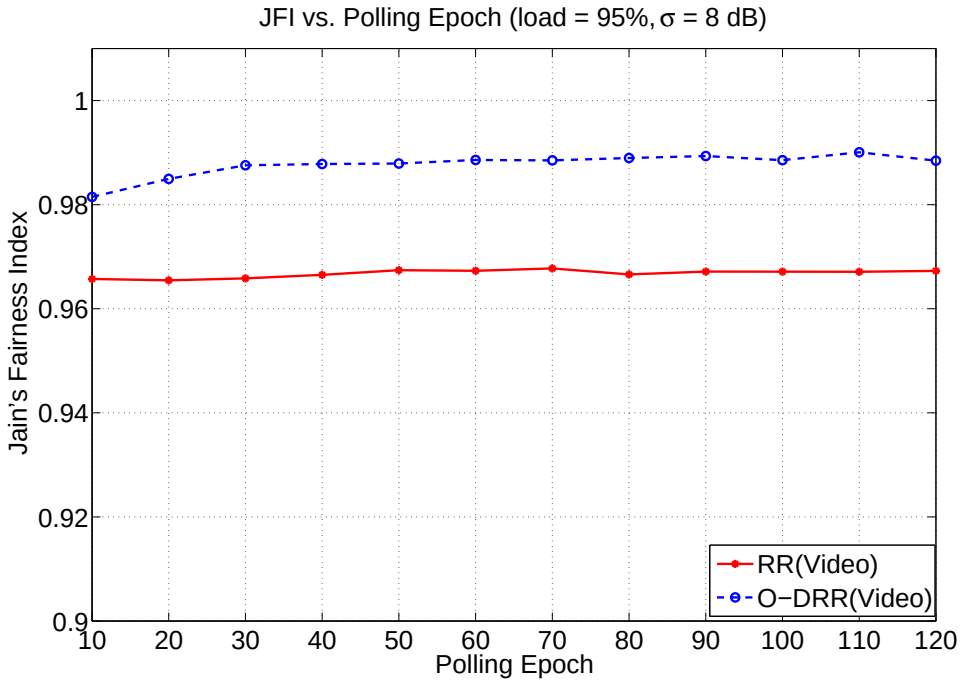


Figure 3.26: Jain's Fairness Index at Different Polling Epochs with Multi-Class Video Traffic

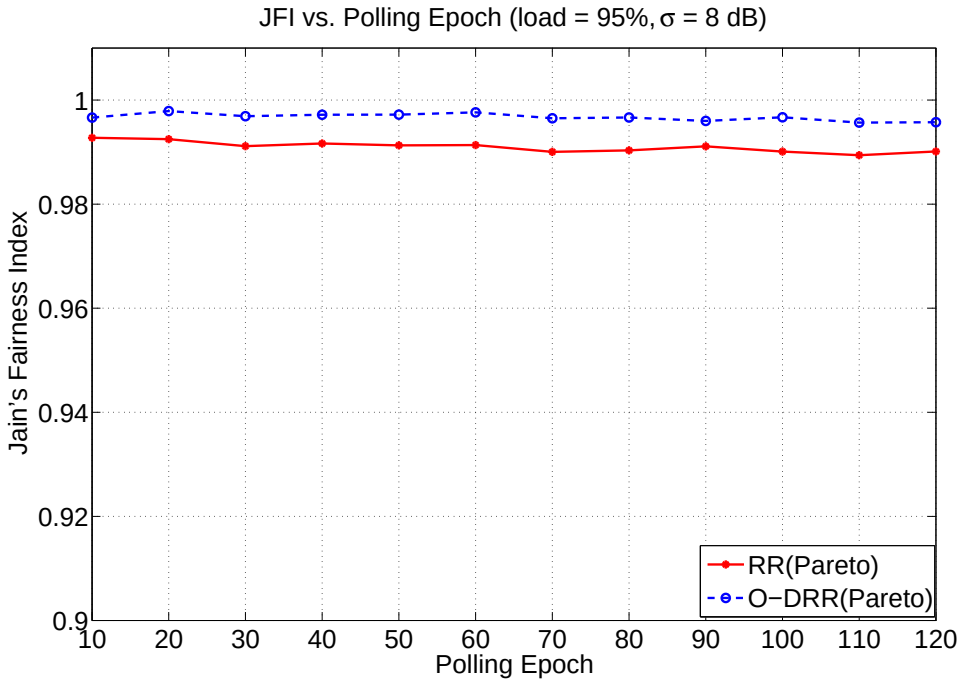


Figure 3.27: Jain's Fairness Index at Different Polling Epochs with Multi-Class Pareto Traffic

Chapter 4

TCP-aware Fair Uplink Scheduling - Fixed Modulation

In the previous chapter, we have considered scheduling of real-time applications. Real-time applications require QoS guarantees in terms of minimum bandwidth and maximum latency from the network. Typically these applications employ UDP as the transport layer protocol. Non real-time Internet applications employ TCP as the transport layer protocol. We term these applications as TCP-based applications. Unlike real-time applications, the TCP-based applications do not operate within the strict guarantee QoS framework. Instead, a TCP source adapts its rate of transmission based on the feedback received from the sink. However, if the underlying scheduling mechanism does not consider this rate adaptation, then it may assign more or less bandwidth than the flows' requirement. If it assigns more bandwidth than required, then bandwidth idleness results. This may lead to aggregate TCP throughput degradation. Similarly, if it assigns less bandwidth than required, then the TCP source will not be able to transmit at its rate resulting in degradation in throughput. This suggests that the scheduler should assign bandwidth based on the requirement of the TCP flows. Since Round Trip Time (RTT) of TCP flows are different, the rate of adaptation of transmission rate of TCP flows are different. Therefore, users with small RTT may grab more bandwidth as compared to users with large RTT. This may lead to unfairness. In wireless networks, bandwidth idleness and throughput degradation become more severe as the channel is time varying in nature. This is because, if the scheduler assigns bandwidth to a flow whose Signal to Noise Ratio (SNR) is not suitable for transmission, then bandwidth allocated to that flow will be not be utilized. We therefore, propose fair TCP

Window-aware Uplink Scheduling (TWUS) and Deadline-based TCP Window-aware Uplink Scheduling (DTWUS) schemes that are cognizant of TCP rate adaptation and the time varying nature of wireless channel.

Though the proposed schemes can be employed in any cellular network, we consider the uplink of multipoint-to-point IEEE 802.16 based WiMAX network in this chapter. As discussed in Chapter 2, the IEEE 802.16 standard defines four service classes such as *UGS*, *rtPS*, *nrtPS* and *BE*. Since TCP-based applications can fall under *nrtPS* and *BE* service classes, the proposed schedulers can be used to schedule *nrtPS* and *BE* service classes. The standard allows both fixed and adaptive modulation schemes in the PHY layer. In this chapter, we consider fixed modulation scheme and in the next chapter we consider adaptive modulation scheme. Using exhaustive simulations, we demonstrate the improvement in TCP throughput achieved by the TCP-aware schedulers over non TCP-aware schedulers. Along with the throughput improvement, we also demonstrate fairness capability and robustness of the scheduling schemes at different channel fading.

In Section 4.1, we explain the characteristics of TCP and the impact of scheduling on TCP performance. In Section 4.2, we discuss the system model and the motivation behind the TCP-aware uplink scheduling schemes. In Section 4.3, we present both TCP Window-aware Uplink Scheduling and Deadline-based TCP Window-aware Uplink Scheduling scheme with fixed modulation. In Section 4.4, we discuss the experimental set up and present the results. These results demonstrate the efficiency of the proposed algorithms. We provide the concluding remarks in Section 4.5.

4.1 Transmission Control Protocol

In this section, we discuss TCP and its characteristics. Though, the contents of this section are standard, we introduce the terminology that has been used in the rest of the chapter. A more comprehensive account of TCP and its characteristics can be found in [42, 43, 72].

TCP is a connection oriented protocol which uses a three-way handshaking mechanism to establish a connection between a source and a sink or destination. It allows the source to transmit back to back packets limited by the congestion window ($cwnd$) size. For every packet the source transmits, it sets a timeout (which known as TCP timeout) before which the source should get an acknowledgement (ACK) for that packet from the sink. The sink transmits ACK

packets for successful reception of either every packet or a group of packets. On successful reception of the ACKs, the source increases its $cwnd$ size and transmits further packets to the sink. However, the data as well as the ACK packets can be lost in the network. To detect the loss of packets, TCP uses two different mechanisms. In the first mechanism, the source assumes packet loss if it receives three duplicate ACKs (*dupacks*) of a single packet in succession. In the second mechanism, the source assumes packet loss if the source does not receive any ACK before the expiry of *TCP timeout*. On detecting a packet loss, the source decreases its $cwnd$ size and retransmits the un-acknowledged packets again. The change in $cwnd$ size from its current value occurs after a minimum duration called Round Trip Time (RTT). In most of the TCP implementations, the value of RTT is estimated by the source using exponential averaging as follows:

$$RTT_{estimated} = \Upsilon \times RTT_{estimated} + (1 - \Upsilon) \times RTT_{measured}, \tag{4.1}$$

where Υ is a constant ($0 < \Upsilon < 1$).

In TCP, instead of using linear increase and decrease of $cwnd$, Additive Increase and Multiplicative Decrease (AIMD) method is employed. Moreover, the manner of increase and decrease of $cwnd$ size is different for different TCP implementations. In most of the TCP implementations, such as TCP Reno and NewReno [73, 74], $cwnd$ drops to one, if the source does not receive an ACK before the TCP timeout. Further, it drops its $cwnd$ to half of its current size, if it receives three *dupacks* in succession.

The increase of $cwnd$ size occurs in two phases. In the first phase, $cwnd$ increases in an exponential manner. This phase is also called as *slow start* phase. In this phase, for every successful ACK that the source receives, TCP increases its $cwnd$ size multiplicatively by a factor of two until it reaches a threshold known as ss_{thresh}. After it reaches ss_{thresh}, it enters the second phase. In this phase, it increases its $cwnd$ size by one for every successful ACK received by it. The increase of $cwnd$ size continues until the source receives three *dupacks* or experiences TCP timeout or it reaches its maximum value $cwnd_{Max}$. Once it reaches $cwnd_{Max}$, it continues transmitting with this window size till it experiences congestion (either it receives triple *dupacks* or experiences TCP timeout). If it receives three *dupacks*, then it updates ss_{thresh} with its new value $\frac{cwnd}{2}$ and enters the linear increase phase. If it experiences TCP timeout, then it drops its $cwnd$ size to one and enters the exponential increase phase again. This process continues till the connection lasts. The $cwnd$ evolution of TCP Reno [73] is depicted in Figure 4.1.

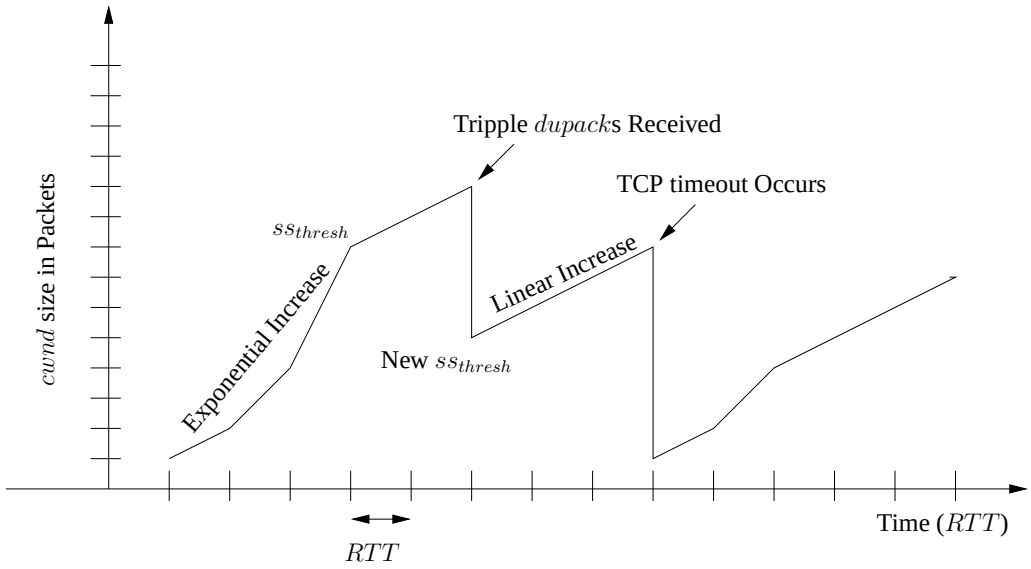


Figure 4.1: Congestion Window Evolution of TCP Reno

As discussed earlier, the actual rate of transmission of TCP-based applications is controlled by the $cwnd$ size. Hence, $cwnd$ size has a significant impact on TCP performance. In the next section, we discuss impact of scheduling on TCP performance.

4.1.1 Impact of Scheduling on TCP Performance

In this section, we discuss the impact of underlying scheduling mechanism on TCP performance. The impact of scheduling on TCP performance can have two major aspects, such as (i) TCP throughput and (ii) fairness. If the scheduling scheme assigns slots without considering the $cwnd$ size of the TCP flows, then the flows with $cwnd = 1$ as well as with $cwnd = cwnd_{Max}$ will be assigned with equal number of slots. For a flow with $cwnd = 1$, slots will be underutilized as its requirement is small, whereas for a flow with $cwnd = cwnd_{Max}$, the number of slots assigned to it may not be adequate to meet its requirement. This results in increase in scheduling delay and may lead to TCP timeouts. Therefore, if the scheduling scheme does not consider the $cwnd$ size of the TCP flows, then TCP throughput suffers and slots get under-utilized,

Like throughput, fairness is also an important parameter of a scheduler. As discussed before, TCP flows with small RTTs increase their $cwnd$ size at a faster rate as compared to the TCP flows with large RTTs. Hence, if the scheduler assigns slots only based on $cwnd$ sizes, then the flows with small RTTs will acquire more number of slots as compared to the flows with relatively large RTTs. Therefore, if the scheduling scheme does not consider RTTs of

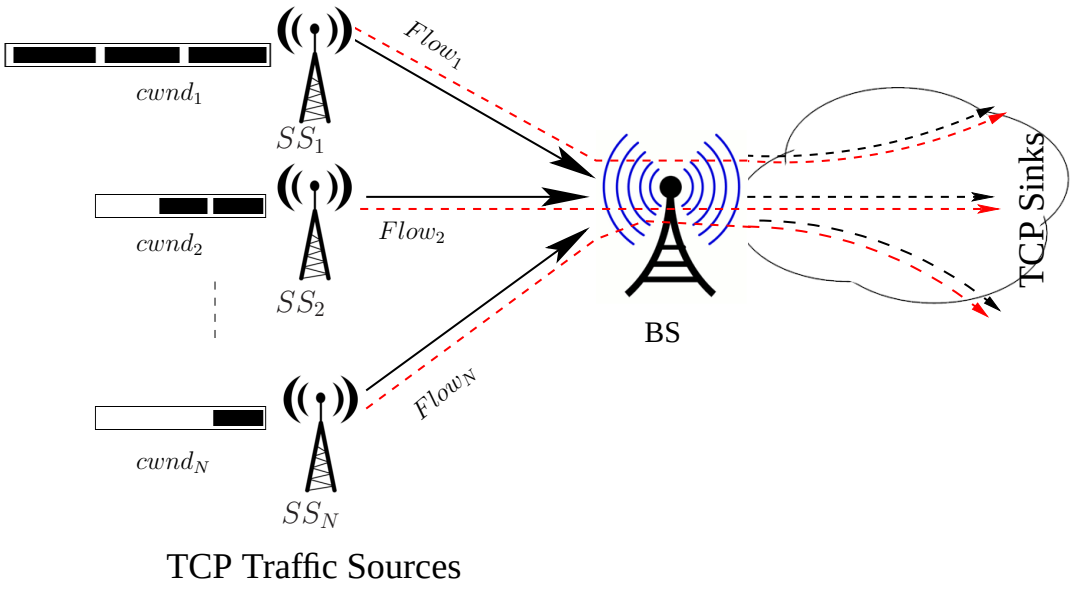


Figure 4.2: Multipoint-to-Point Framework with TCP-based Applications

the TCP flows, it may result in unfairness among the flows.

4.2 System Model

We consider a multipoint-to-point network where multiple SSs are connected to one BS as shown in Figure 4.2. This scenario may correspond to a single cell IEEE 802.16 [3] system. Though some part of this model are similar to that discussed in the previous chapter, we re-state the important assumptions required for this chapter.

BS is the centralized entity responsible for scheduling the TCP flows. We assume that the SSs are TCP traffic sources. Each packet is associated with a TCP flow (also known as source-sink pairs) and each flow is associated with a SS. Though this can be generalized to multiple flows per SS, we consider a single flow per SS. TCP acknowledgement (ACK) packets traverse from the sink to the source in the downlink direction. We further assume that the ACK packets are very small and the downlink scheduler at the BS schedules these ACK packets without any delay.

Time is divided into frames. Each frame (of duration of T_f) in turn is composed of a fixed number of slots of equal duration of T_s. We assume time varying wireless channel between a SS and the BS. We assume that the channel gains between SSs and the BS are independent and identically distributed (i.i.d.). We further assume that the coherence time of the channel is

greater than the frame duration, i.e., the channel state remains constant during a frame duration. The channel state changes from frame to frame according to Rayleigh fading model. We also consider Log-normal shadowing and path loss in modelling the channel gain. We assume that the individual channel state information is available at the BS in every frame. Let SNR_i denote the SNR measured between SS_i and the BS. Packets can be successfully received if $SNR_i \geq SNR_{th}$. The value of SNR_{th} depends upon the modulation and coding scheme used at the Physical (PHY) layer. Since we consider fixed modulation technique in this chapter, the maximum attainable data rate is also fixed.

We assume that a set I of TCP flows shares a network of I unidirectional links through the BS[1]. The capacity of the individual link i denoted by c_i, for $i = 1, 2, 3, ...I$, is a function of the SNR of the corresponding link. c_i can also be considered as maximum attainable data rate at link i. Since the channel state varies from frame to frame, c_i also varies from frame to frame.

4.2.1 Motivation and Problem Formulation

In this section, we motivate for a centralized polling based uplink scheduling algorithm for TCP-based applications. As pointed out in Chapters 1 and 2, traditional scheduling schemes like Weighted Fair Queuing (WFQ) [26], Self-Clocked Fair Queueing (SCFQ) [27] and Worst-case Fair Weighted Fair Queuing (WF^{2}Q) [28], etc., cannot be used for the uplink scheduling. Instead, variants of Round Robin (RR) schedulers are the candidates for uplink scheduling. Since the channel state of SSs varies randomly across the frames, a RR scheduler would result in unfairness. Moreover, RR scheduler does not consider $cwnd$ size, RTT and TCP timeout while scheduling. As discussed in Section 4.1.1, since TCP throughput as well as fairness suffers if the underlying scheduling scheme does not consider $cwnd$ size and RTT of the flows, a simple RR scheduler is not appropriate. Therefore, we propose TCP-aware scheduling mechanisms, which attempt to schedule flows based on $cwnd$ size and TCP timeouts as well as maintain fairness among the flows.

For the centralized uplink scheduling, the SSs are required to communicate their bandwidth requirements with the BS either in a contention mode or in a polling mode. Since $cwnd$ size of a SS does not change for one RTT, the rate of transmission $x = \frac{cwnd}{RTT}$ remains constant for one RTT. Hence, bandwidth requirement of any SS does not change for one RTT. In prac-

[1] We assume one to one mapping between the flows and the links.

tice, one RTT spans over 150 to 200 msec, whereas one frame duration is of the order of 1 to 5 msec. Hence, contention based scheduling, where bandwidth request is communicated in every frame, is not appropriate. Instead, a polling based solution, where bandwidth requirement is communicated after certain number of frames (called polling epoch) is more appropriate. Since the resource requirement changes only after one RTT, selecting a polling epoch of duration less than one RTT is not appropriate.

Unlike wired network, the time varying nature of wireless channel and the modulation techniques used at the PHY layer also have an impact on scheduling. As discussed before, we consider fixed modulation in this chapter. In addition, we exploit the idea of Opportunistic scheduling while designing these systems. Since we use both $cwnd$ size and TCP timeout in the scheduling process, we classify the scheduling schemes into (i) TCP Window-aware Uplink Scheduling (TWUS) with Fixed Modulation; and (ii) Deadline-based TCP Window-aware Uplink Scheduling (DTWUS) with Fixed Modulation. We discuss the details of the scheduling algorithms in the subsequent sections.

4.3 TCP-aware Uplink Scheduling with Fixed Modulation

Before discussing the scheduling algorithm, we define the following terms. Though the terms defined here are similar to the terms defined in the previous chapter, we re-state them in the context of TCP-aware scheduling.

- *Connected Set*: The set of SSs that has been admitted into the system through an admission control and connection set up phase is called connected set ($L_{connect}$). Let N be the cardinality of the connected set.

- *Polling Epoch*: It is defined as the interval that the BS chooses to poll the connected SSs. In the proposed scheduling algorithm, the polling is performed by the BS only once after every k frame.

- *Schedulable Set*: A SS is schedulable, if at the beginning of a polling epoch, it has a non zero $cwnd$ size and the SNR of its wireless link to the BS is above a minimum threshold SNR_{th}. The set of such SSs constitute a schedulable set L_{sch}. This set may change dynamically across the polling epochs. Let M be the cardinality of the schedulable set.

- *Active Set*: An *active* SS is defined to be one that is schedulable during a given frame of a polling epoch and was schedulable at the beginning of that epoch. The set of such SS constitutes an active set L_{active}. During a polling epoch, the BS only schedules traffic from the active set. The membership of an active set may change dynamically across the frames, whereas the membership of a schedulable set may change across the polling epochs.

- *Quantum Size*: It is the number of slots that should be assigned to any of the schedulable SS in any frame of a polling epoch. At the beginning of every polling epoch, the BS determines the quantum size as: $Q = \frac{N_s}{M}$, where N_s is the total number of uplink slots available for scheduling. Since M is fixed for a polling epoch, Q is also fixed for a polling epoch.

We divide the proposed scheduling algorithm into two phases: *polling* and *slot assignment*. The periodic polling epoch k is a function of RTTs of the contending flows. We discuss the determination of polling epoch k in the next section in detail. BS polls all connected SSs after every k frames. Once the polling is performed, the BS schedules the active SSs in every frame. The relationship between polling epoch and frame by frame scheduling is illustrated in Figure 4.3.

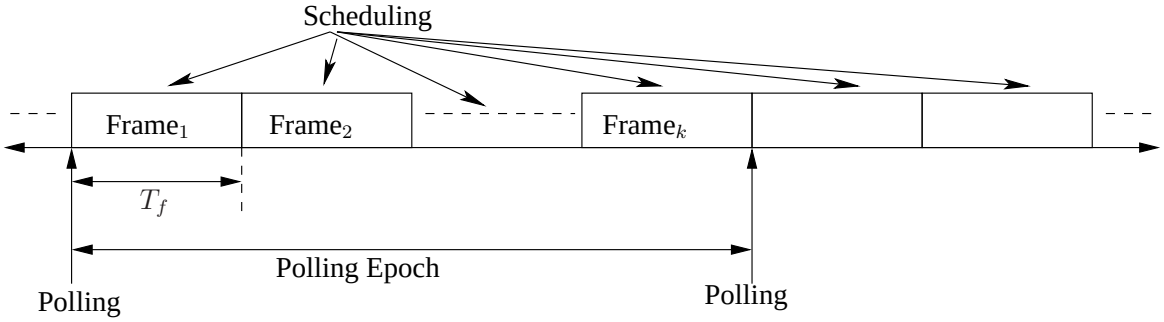


Figure 4.3: Polling and Frame by Frame Scheduling in TCP-aware Scheduling

Let PL denote the length of a packet in bits. The number of slots required to transmit $cwnd$ number of packets at rate R in bits/sec is expressed as:

$$n_i^{req} = \frac{cwnd_i \times PL}{R \times T_s}. \tag{4.2}$$

n_i^{req} is the total number of slots required by SS_i to transmit $cwnd_i$ number of packets in one RTT duration. However, both the channel gains and the number of slots assigned to

each SS are different from frame to frame. Therefore, the amount of data transmitted by each SS differs from frame to frame. Hence, the determination of n_i^{req} at the beginning of polling epoch is not appropriate. Instead, there is a need to determine the number of slots required using the resource requirement and capacity of the link in every frame. Resource requirement is determined by the scheduler by determining

$$\begin{aligned} D_i(0) &= cwnd_i \times PL, \\ D_i(n) &= cwnd_i \times PL - Tx_i(n-1), \forall n \in [1,k], \end{aligned} \tag{4.3}$$

where $Tx_i(n-1)$ is the total amount of data (in bits) transmitted by SS_i from the beginning of the current polling epoch to the frame under consideration. At the beginning of a polling epoch, the amount of data transmitted by a SS is set to zero, i.e., $Tx_i(0) = 0, \forall i \in I$. The number of slots required by SS_i in n^{th} frame is given by:

$$n_i^{req}(n) = \frac{D_i(n)}{R \times T_s}. \tag{4.4}$$

At the time of polling, each polled SS needs to communicate its $cwnd$ in terms of the number of slots required and its RTT with the BS. In the subsequent sections, we first discuss the method to chose the polling epoch and then discuss the method of slot assignment by the BS. Note that, like O-DRR scheduler, TCP-aware schedulers require flow level information only.

4.3.1 Determination of Polling Epoch k

The polling epoch k is an important parameter of the proposed scheduler. k should be chosen in such a manner that the system efficiency and fairness are maintained. We argue that the polling epoch k should be the minimum RTT[2] among all TCP flows going through the BS. This is because, the TCP timeout value is typically chosen to be four to five times the RTT in most TCP implementations. Therefore, if we choose the polling epoch to be equal to two RTTs, then any SS with an ongoing TCP flow that misses polling needs to be polled at the next opportunity. Similarly, if the polling epoch is more than two RTTs, and if the BS misses one SS with a

[2]Typical TCP RTTs are in the range of 100 msec - 200 msec, whereas the frame length T_f in IEEE 802.16 is 0.5 msec, 1 msec or 2 msec.

TCP flow, then congestion window reduction for that TCP flow is likely to occur with high probability. This is because, the chances of not getting scheduled at the next opportunity before TCP timeout is very high. If polling is very frequent, i.e., more than once per RTT, then more control slots will be utilized for polling. Moreover, frequent polling is not preferable, since the $cwnd$ size itself changes after one RTT. Hence, we choose the polling epoch to be equal to the $\min\{RTT\}$ of the active TCP flows. In the proposed scheduling scheme, after every polling epoch, the BS determines the next polling epoch based on the $\min\{RTT\}$received from the polled SSs of the current polling epoch.

4.3.2 Slot Assignments using TCP Window-aware Uplink Scheduling

BS maintains an indicator variable $Flag_i$ for each SS. $Flag_i$ is 1, if SS_i is scheduled in a frame, and 0 otherwise. Let $N_i(n)$ be the total number of slots assigned to SS_i in frame n. Based on the number of slots assigned in the previous frame and $cwnd$ size, the BS determines the resource requirement for each schedulable SS in every frame. $D_i(n)$ (as defined in Eqn. (4.3)) of a schedulable SS in frame n is updated as:

$$\begin{aligned} D_i(n) &= cwnd_i \times PL - Tx_i(n-1) \\ &= D_i(n-1) - Flag_i(n-1) \times N_i(n-1) \times R \times T_s, \forall n \geq 1, \end{aligned} \tag{4.5}$$

where R is the rate of transmission between SSs and the BS. In this case R is fixed. At the beginning of a polling epoch, resource requirement of each SS is equivalent to the number of slots required to transmit its $cwnd$ size, i.e., $D_i(0) = cwnd_i \times PL$. In a polling epoch, if $D_i(n)$ of SS_i becomes zero, then SS_i is withdrawn from the active set.

We utilize DRR's idea of maintaining a quantum size Q for each schedulable SS. The quantum size Q is kept fixed for one polling epoch. The number of slots assigned to any subscriber station SS_i in frame n can be greater or smaller than the quantum size. To keep a track of the number of slots assigned with respect to the quantum size Q, we use a credit-based approach. The credit-based approach employs one deficit counter for each SS_i similar to DRR. The idea of a deficit counter is to ensure fairness among the subscriber stations in long term. At the beginning of a polling epoch, deficit counter of SS_i is initialized to one. The deficit counter $DC_i(n)$ is updated in every frame as:

$$\begin{aligned} DC_i(0) =& 1, \\ DC_i(n) =& DC_i(n-1) + Q - Flag_i(n-1) \times N_i(n-1),\ \forall i \in L_{sch}, \forall n \geq 1. \end{aligned} \tag{4.6}$$

From the above equation, we observe that the deficit counters of all schedulable SSs are incremented by the quantum Q, whereas the deficit counter of all active SSs are decremented by the amount of slots received in the previous frame. We define scaled deficit counter dc_i as follows:

$$\begin{aligned} dc_i(0) =& 1,\ \forall i \in L_{active}, \\ dc_i(n) =& DC_i(n) + \min_j |DC_j(n)|,\ \forall i, j \in L_{active},\ \forall n \geq 1. \end{aligned} \tag{4.7}$$

At the beginning of a polling epoch, the scaled deficit counter dc_i of SS_i is initialized to one. After determining the resource requirement and scaled deficit counter, the BS determines the weight of each active set SS. For all other SSs, the weights are zero. BS determines the weight $W_i(n)$ for SS_i in frame n using the following equation:

$$W_i(n) = \frac{D_i(n) \times dc_i(n)}{\sum_{j \in L_{active}} D_j(n) \times dc_j(n)}, \forall i \in L_{active},\ \ \forall n \geq 1. \tag{4.8}$$

Eqn. (4.8) essentially determines a weight $W_i(n)$ for an SS_i in frame n, which is proportional to the normalized product of the scaled deficit counter and resource requirement (and hence the $cwnd$). If a TCP flow has a smaller RTT, its $cwnd$ size will be increased at a faster rate. Since we chose $k = \min_i\{RTT_i\}$, at the beginning of each polling epoch, SS with the smallest RTT will update its $cwnd$ size, whereas others will not. This will result in increase in resource requirement of a SS whose RTT is small as compared to other SSs. Hence, by allocating the number of slots in proportion to the resource requirement, will result in allocation of larger share of slots to such flows. This will also result in unfairness among the SSs. The credit-based approach proposed by us ensures that the scaled deficit counter value is small for such flows and thereby ensures fairness. After the determination of weights, the BS assigns the number of slots to SS_i in frame n using:

$$N_i(n) \ \ = \ \ \frac{W_i(n) \times N_s}{\sum_{j \in L_{active}} W_j(n)}, \forall i \in L_{active},\ \ \forall n \geq 1. \tag{4.9}$$

In the next section, we exploit the knowledge of TCP timeout along with the congestion window size information and propose another scheduling mechanism. We call this scheduling algorithm as Deadline-based TCP Window-aware Uplink Scheduling (DTWUS).

4.3.3 Slot Allocation using Deadline-based TCP Window-aware Uplink Scheduling

In this scheme, the determination of deficit counter, scaled deficit counter and resource requirement for each SS is similar to that defined in TWUS scheduler. However, the method of determination of weights for scheduling is different from that defined in Eqn. (4.8). In this scheme, we use TCP timeout information along with resource requirement and deficit counter values to determine the weights. In this scheme, at the beginning of the polling, each polled SS communicates its $cwnd$ size, time left to reach TCP timeout (TTO) (of the last un-acknowledged packet), TCP timeout and RTT to the BS.

For each schedulable SS, we define *deadline* d_i to be a measure of its rate of approach towards the TCP timeout. At the beginning of a polling epoch, let TTO_i denote the maximum duration that the TCP flow of SS_i can wait before reaching TCP timeout. Note that the maximum value of TTO_i is the TCP timeout[3] associated with the TCP flow belongs to SS_i. At the beginning of a polling epoch, d_i of SS_i is initialized to TTO_i. If SS_i is scheduled in frame n, then the deadline $d_i(n)$ remains unchanged, i.e, it takes the value of $d_i(n-1)$. Otherwise, $d_i(n)$ is decremented by one frame duration from its previous value. BS updates the deadline of the schedulable flows as follows:

$$\begin{aligned} d_i(0) =& TTO_i,\ \forall i \in L_{connected}, \\ d_i(n) =& d_i(n-1) - T_f,\ \forall i \in (L_{sch} \setminus L_{active}),\ \forall n \geq 1, \\ d_i(n) =& d_i(n-1),\ \forall i \in L_{active},\ \forall n \geq 1. \end{aligned} \tag{4.10}$$

If T_f exceeds $d_i(n)$, then the deadline $d_i(n)$ of SS_i is initialized to TCP timeout (TO_i) of that SS. In that case, TCP flow experiences a timeout before getting scheduled, resulting in reduction of $cwnd_i$ to one. After determining the scaled deficit counter as in Eqn. (4.7) and deadline as in Eqn. (4.10), the BS determines weight $W_i(n)$ for SS_i in frame n using:

[3]TCP flows generally start at random and hence different flows have different residual times to reach TCP timeout.

$$W_i(n) = \frac{\frac{D_i(n) \times dc_i(n)}{d_i(n)}}{\sum_{j \in L_{active}} \frac{D_j(n) \times dc_j(n)}{d_j(n)}}, \forall i \in L_{active}, \ \ \forall n \geq 1. \tag{4.11}$$

After the determination of weights, the number of slots assigned by the BS to SS_i in frame n is determined as:

$$N_i(n) \ = \ \frac{W_i(n) \times N_s}{\sum_{j \in L_{active}} W_j(n)}, \forall i \in L_{active}, \ \ \forall n \geq 1. \tag{4.12}$$

The use of deadline in weight determination ensures that a higher number of time slots are assigned to a SS that has a smaller deadline.

The pseudo-code of the proposed schedulers TWUS and DTWUS is presented in Algorithm 2. We combine both schedulers by using $Flag_{deadline}$, which is set to one for DTWUS and is set to zero for TWUS. In the next section, we discuss the experimental setup and simulation results in an IEEE 802.16 setting.

4.4 Experimental Evaluation of TCP-aware Schedulers

In this section, we describe simulation experiments that have been performed to evaluate TCP-aware scheduling. All the simulations have been conducted using implementations of TCP-aware scheduling with IEEE 802.16 setting in MATLAB [68]. We consider a multipoint-to-point IEEE 802.16 based network where 10 SSs are connected to a centralized BS as shown in Figure 4.2. We simulate one TCP flow per SS. Each TCP flow starts randomly. The RTTs of the flows are updated using exponential averaging. Each SS is assumed to have a large enough buffer at its interface, such that the probability of buffer overflow is negligible. The frame duration T_f is set equal to 2 msec[4]. The uplink subframe T_{ul} consists of 500 data slots (we assume that the number of control slots used is negligible). We consider both equal and unequal distances between SSs and the BS. For equal distances, the distances of all SSs from the BS are 1 km each and for unequal distances, the distances between SSs (SS_1 - SS_{10}) and the BS are 0.857 km, 1.071 km, 0.910 km, 1.230 km, 1.113 km, 0.956 km, 1.122 km, 0.884 km, 0.970 km and 1.216 km respectively (given in Table 4.1).

We further consider WirelessMAN-SC air interface for the simulations. We implement QPSK modulation scheme at the PHY layer. We consider $SNR_{th} = 12.18$ dB, the minimum

[4]Frame duration (T_f) is equally divided between uplink subframe (T_{ul}) and downlink subframe (T_{dl}).

Algorithm 2 :TCP-aware Uplink Scheduler with Fixed Modulation

while TRUE **do**
  Determine L_{sch} for the current polling epoch
  $Flag_i(0) \leftarrow 0 \; \forall i \in L_{sch}$
  $D_i(0) \leftarrow cwnd_i \times PL \; \forall i \in L_{sch}$
  $DC_i(0) \leftarrow 1, \; dc_i(0) \leftarrow 1 \; \forall i \in L_{sch}$
  $W_i(0) \leftarrow 0, \; N_i(0) \leftarrow 0 \; \forall i \in L_{sch}$
  if $Shceduler_{Type} = TWUS$ **then**
    $Flag_{deadline} = 0, \;\; d_i(0) \leftarrow 1 \; \forall i \in L_{sch}$
  else
    $Flag_{deadline} = 1, \;\; d_i(0) \leftarrow TTO_i \; \forall i \in L_{sch}$
  end if
  $M \leftarrow |L_{sch}|, \; Q \leftarrow \frac{N_s}{M}$
  $k \leftarrow \min_i\{RTT_i\}, \; T \leftarrow k \times T_f$
  Frame number $n \leftarrow 1$
  while $T > 0$ **do**
    $L_{active} \leftarrow \phi$
    for all $i \in L_{sch}$ **do**
      if $(SNR_i(n) \geq SNR_{th}) \; \Lambda (D_i(n-1) > 1)$ **then**
        $L_{active} \leftarrow L_{active} \cup \{i\}$
        $Flag_i(n) \leftarrow 1$
        $DC_i(n) \leftarrow DC_i(n-1) + Q - Flag_i(n-1) \times N_i(n-1)$
        if $Flag_{deadline}$ = 1 **then**
          $d_i(n) \leftarrow d_i(n-1)$
        else
          $d_i(n) \leftarrow 1$
        end if
      else
        $Flag_i(n) \leftarrow 0$
        $DC_i(n) \leftarrow DC_i(n-1) + Q$
        $D_i(n) \leftarrow D_i(n-1)$
        if $Flag_{deadline}$ = 1 **then**
          $d_i(n) \leftarrow d_i(n-1) - T_f$
        else
          $d_i(n) \leftarrow 1$
        end if
        if $d_i(n) \leq 0$ **then**
          $d_i(n) \leftarrow TO_i$
        end if
        $W_i(n) \leftarrow 0, N_i(n) \leftarrow 0$
      end if
    end for
    for all $i \in L_{active}$ **do**
      $D_i(n) \leftarrow D_i(n-1) - Flag_i(n-1) \times N_i(n-1) \times R \times T_s$
      $dc_i(n) \leftarrow DC_i(n) + \min_j |DC_j(n)|, \forall j \in L_{active}$
      $W_i(n) \leftarrow \frac{\frac{D_i(n) \times dc_i(n)}{d_i(n)}}{\sum_{j \in L_{active}} \frac{D_j(n) \times dc_j(n)}{d_j(n)}}$
      $N_i(n) \leftarrow \frac{W_i(n) \times N_s}{\sum_{j \in L_{active}} W_j(n)},$
    end for
    $T \leftarrow T - T_f, \; n \leftarrow n + 1$
  end while
end while

Table 4.1: Distance (d) between SSs and the BS (in km)

Equal	**Unequal**									
1.0	0.857	1.071	0.910	1.230	1.113	0.956	1.122	0.884	0.970	1.216

SNR required for IEEE 802.16-2004 WirelessMAN-SC air interface in our simulations. The path loss exponent due to distance is set as $\gamma = 4$. We simulate both shadowing as well as fast fading in our experiments. We also consider AWGN with PSD N_0 = 0.35 (4.5 dB/Hz). The shadowing is modeled as Log-normal with mean zero and standard deviation (σ) of 8 dB. In each simulation run, the channel gain due to Log-normal shadowing is kept fixed for a duration of 50 frames. For fast fading, we consider Rayleigh fading model. The channel gain due to fast fading is modeled as complex Gaussian random variable or equivalently the power gain is an exponential random variable with mean β. The coherence time of the channel is considered to be equal to one frame duration, i.e, the channel gain due to fast fading changes from frame to frame. The value of β and transmission power is chosen such that the expected SNR received at the cell edge is more than SNR_{th} required for transmission. We also repeat the experiments with different Log-normal shadowing with σ of 4, 6, 8, 10 and 12 dB.

We conduct four sets of experiments based on distance (equal and unequal) and the proposed schedulers (TWUS and DTWUS). For each set of experiment, we vary σ of Log-normal shadowing as discussed above. The system parameters used for simulations are presented in Table 4.2. The value of each performance parameter observed has been averaged over 50 independent simulation runs.

4.4.1 Simulation Results

Impact of $cwnd_{Max}$

Since $cwnd_{Max}$ value controls the TCP throughput, choosing its correct value in simulations is very important. A very high $cwnd_{Max}$ will cause more congestion and packet drops due to buffer overflow, whereas a small $cwnd_{Max}$ will under-utilize the network. Hence, before conducting the experiments to verify the performance of the proposed schedulers, we perform experiments to measure the average TCP throughput at different $cwnd_{Max}$ values. In Figure 4.4, we plot the average TCP throughput achieved vs. $cwnd_{Max}$ at σ = 8 dB. From this figure, we observe that TCP throughput remains constant (reaches saturation) once the $cwnd_{Max}$ reaches 60 packets. We choose $cwnd_{Max}$ = 60 for the rest of our experiments in this chapter.

Table 4.2: Summary of System Parameters

Simulation Parameter	**Value**
Channel Bandwidth	25 MHz
Modulation Scheme	QPSK
Path Loss Exponent (γ)	4
Frame Length T_f	2 msec
Uplink/Downlink Frame Length	1 msec
Number of Data Slots per T_{ul}	500
Number of Frames Simulated	40000
TCP Type	TCP Reno
Number of Independent Runs	20
No. of SSs	10
Packet Size	8000 bits

4.4.2 Comparison With Weight-based Schedulers

We compare the performance of TCP-aware scheduler with that of a non-TCP aware scheduler, namely, the Weight-based (WB). We implement both Channel Dependent demand driven (WB (CD)) and Channel Independent demand driven Weight-based (WB (CI)) schedulers at the BS for the uplink flows. We assume equal weights for each SS for the implementation and determine the resource requirement of each user using Eqn. (4.5). The polling is employed to determine the schedulable list at the beginning of the polling epoch for both WB (CI) and WB (CD) schedulers. The polling epoch k used in WB schedulers is same as that used in the TCP-aware schedulers. BS schedules the *active* SSs using WB (CI) and WB (CD) schedulers. We determine the average $cwnd$ size, average TCP throughput, slot utilization and Jain's Fairness Index (JFI) achieved by each of the schedulers for a fair comparison.

Average $cwnd$ size Comparison

In Figures 4.5 - 4.6, we plot the average $cwnd$ size achieved by the WB schedulers and TCP-aware schedulers under different standard deviation (σ) of Log-normal shadowing with equal and unequal distances respectively. From these figures, we observe that as σ of Log-normal shadowing increases, the average $cwnd$ size achieved by both WB and TCP-aware schedulers

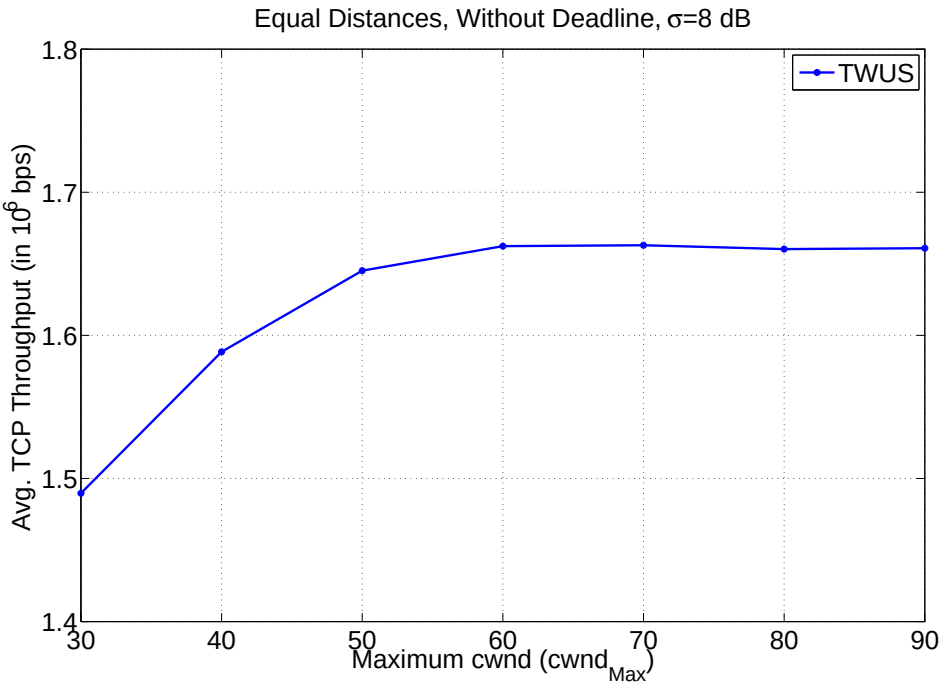


Figure 4.4: Average TCP Throughput vs. $cwnd_{Max}$, with Fixed Modulation

decreases. We also observe that the average $cwnd$ size achieved by the TCP-aware schedulers are higher than that of WB schedulers for all shadowing cases. In Figure 4.7, we plot the percentage increase in the average $cwnd$ size by the TCP-aware schedulers over WB schedulers under different shadowing. From this figure, we observe that as the σ of Log-normal shadowing increases, the gain in $cwnd$ size also increases. The gain in average $cwnd$ size of TWUS over WB (CD) varies between 21% to 34%, whereas it varies between 37% to 88% for TWUS over WB (CI). Though we have illustrated the results for equal distance experiments, similar comparisons are also valid for unequal distance experiments. We also observe that the average $cwnd$ size achieved by the DTWUS is more than that of TWUS. As discussed before, by using TCP timeouts in scheduling, the chances of reducing the $cwnd$ size to one is minimized. This results in increase in average $cwnd$ size.

We also compare the performance of WB (CD) and WB (CI) schedulers. From Figures 4.5 - 4.6, we observe that the average $cwnd$ size achieved by the WB (CD) scheduler is more than that of WB (CI) scheduler, irrespective of distances. Since WB (CD) scheduler is channel dependent, the slots allocated to SSs are better utilized. This ensures less slot idleness in WB (CD) as compared to WB (CI) scheduler. Therefore, the average $cwnd$ size achieved in WB (CD) scheduler is more than that of WB (CI) scheduler.

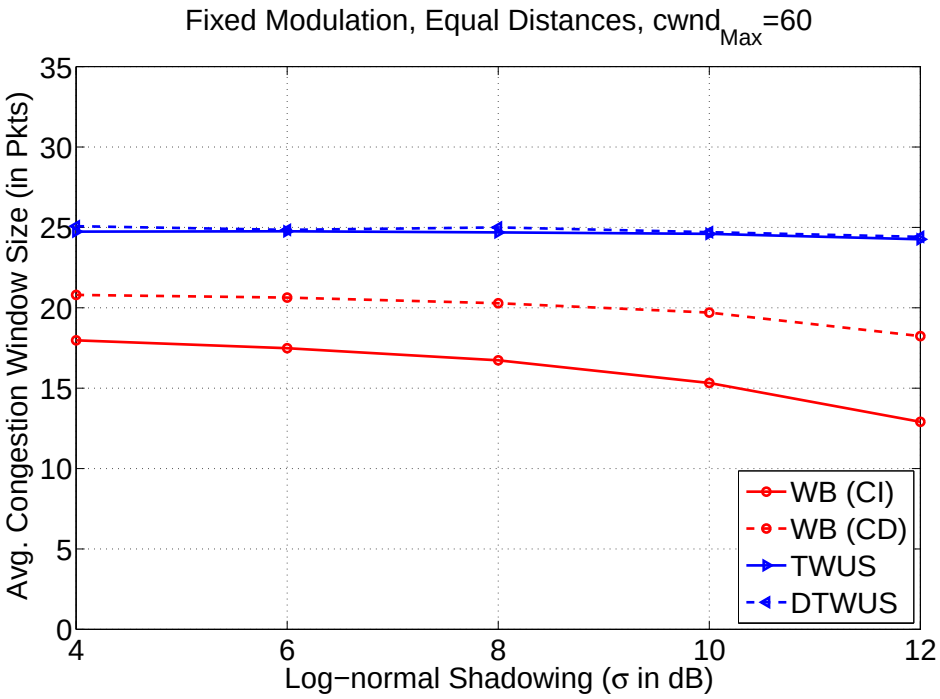


Figure 4.5: Average $cwnd$ size Comparison - Fixed Modulation, Equal Distances

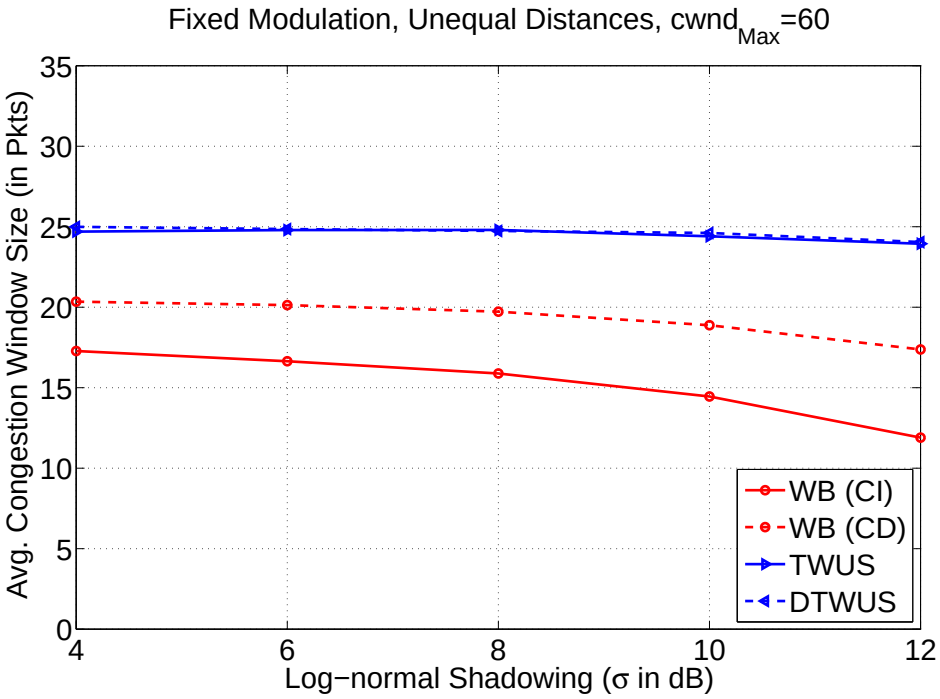


Figure 4.6: Average $cwnd$ size Comparison - Fixed Modulation, Unequal Distances

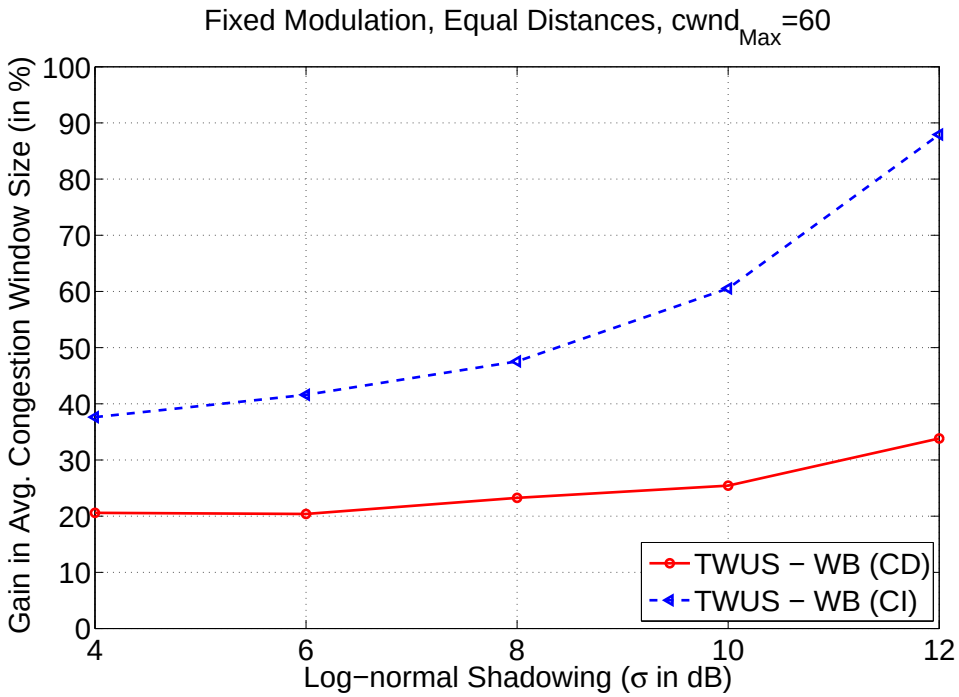


Figure 4.7: Gain in $cwnd$ size of TWUS over WB Schedulers - Equal Distances

Average Throughput Comparison

We also compare the average TCP throughput achieved by the schedulers. For TCP throughput determination, we consider the number of successful transmitted packets only and do not consider retransmissions. In Figures 4.8 - 4.9, we plot the average TCP throughput obtained by the TCP-aware schedulers and WB schedulers under different shadowing and distances. We also plot the gain in average TCP throughput achieved by the TCP-aware scheduler over the WB schedulers in Figure 4.10. The improvements observed in TCP throughput is similar to that of the improvement in average $cwnd$ size illustrated in Figure 4.7. We also observe that the gain in average TCP throughput of TWUS over WB (CI) varies between 39% to 97%, whereas it varies between 24% to 42% for TWUS over WB (CD).

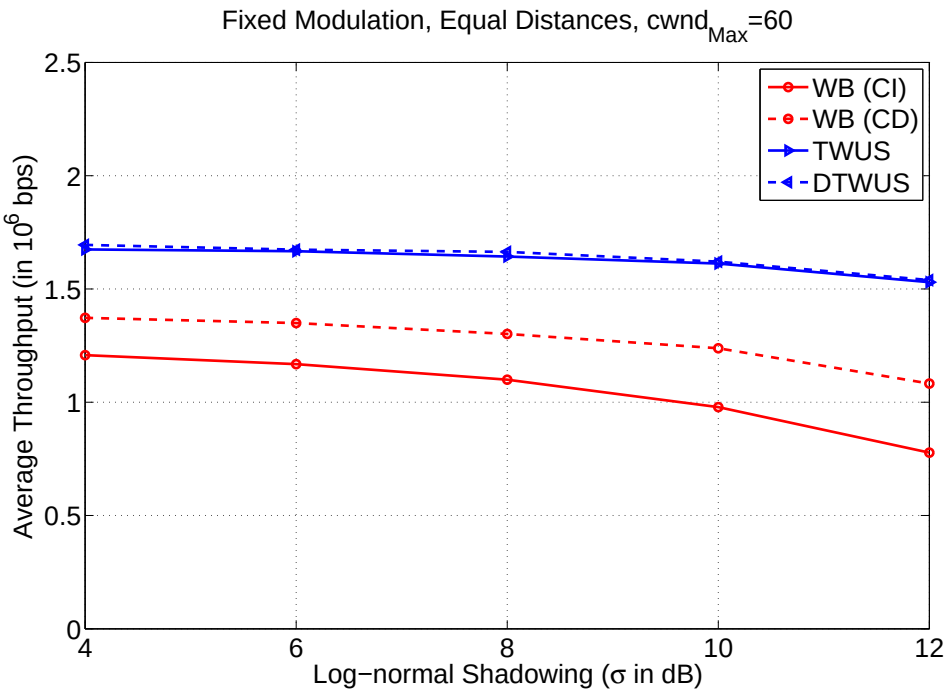


Figure 4.8: TCP Throughput Comparison - Fixed Modulation, Equal Distances

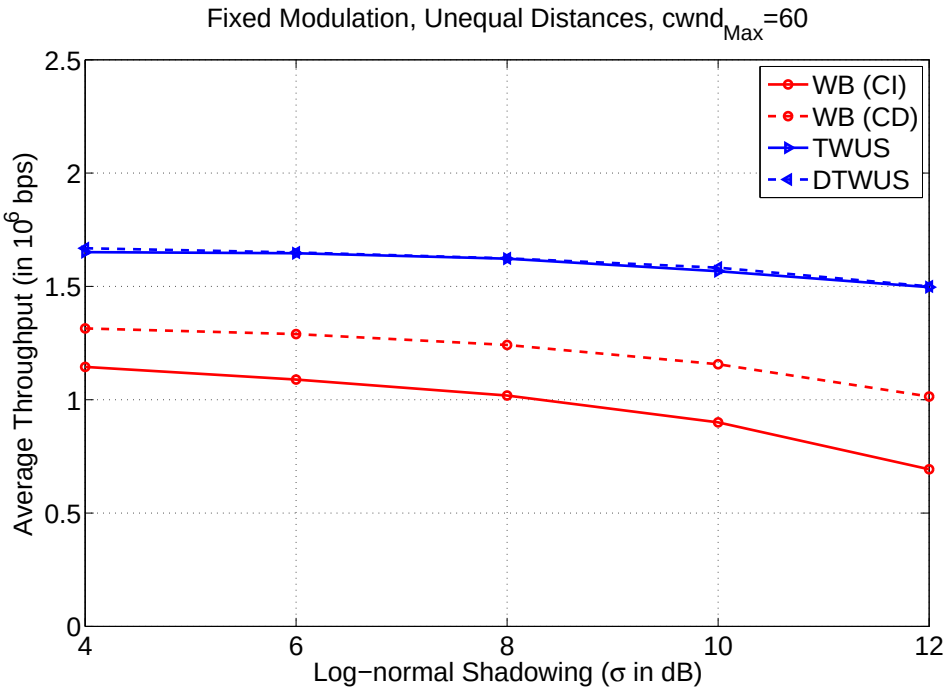


Figure 4.9: TCP Throughput Comparison - Fixed Modulation, Unequal Distances

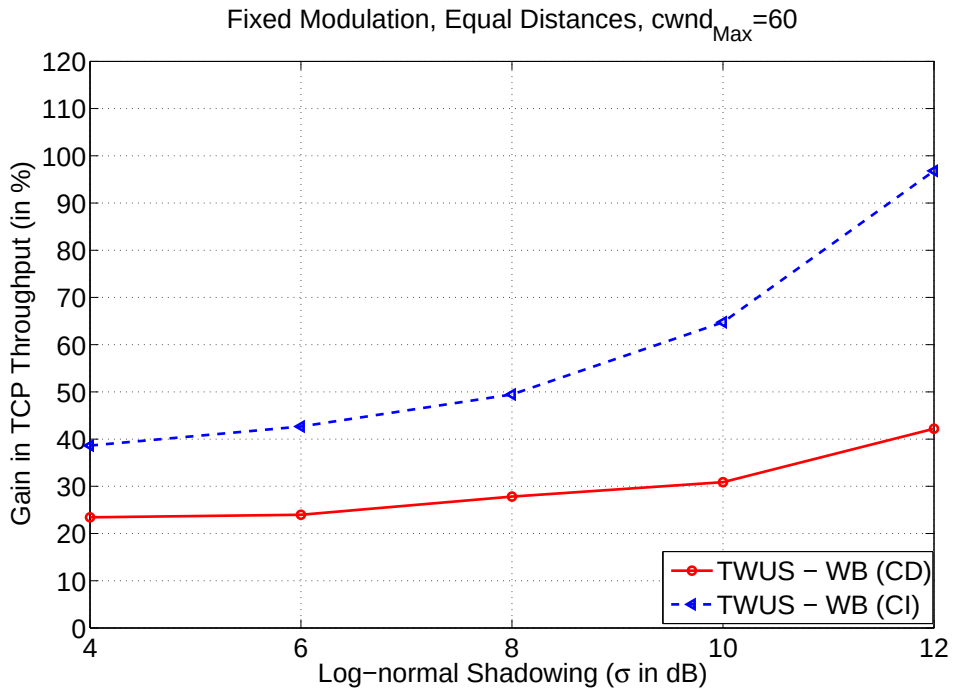


Figure 4.10: Gain in TCP Throughput of TWUS over WB Schedulers - Equal Distances

Fairness Comparison

To compare the fairness of TCP-aware schedulers with that of WB schedulers, we determine Jain's Fairness Index (JFI) for the amount of data transmitted by each SS considering equal and unequal distances between SSs and the BS. We plot the variation of JFI thus determined for different values of σ of the Log-normal shadowing in Figures 4.11 - 4.12. From these figures, we observe that the JFI is higher than 91% under all shadowing cases. JFI of TCP-aware schedulers are close to 99% in the equal distance experiment, whereas it varies between 91% to 98% in the unequal distance experiment. This illustrates that the TCP-aware scheduling schemes are fair under various channel fading conditions. We also observe that the JFI of both TCP-aware and WB schedulers remains constant under different σ of Log-normal shadowing. We observe that TCP-aware schedulers outperform WB schedulers with JFI as performance metric. JFI obtained by the TCP-aware schedulers are at least 5% above that obtained by the WB (CI) scheduler, whereas it is very close to that achieved by WB (CD) scheduler. Since the WB (CI) scheduler is channel independent, the JFI achieved by this scheduler is low as compared to other schedulers. Moreover, unlike other schedulers (TWUS, DTWUS and WB (CD)), JFI of WB (CI) decreases as the shadowing increases.

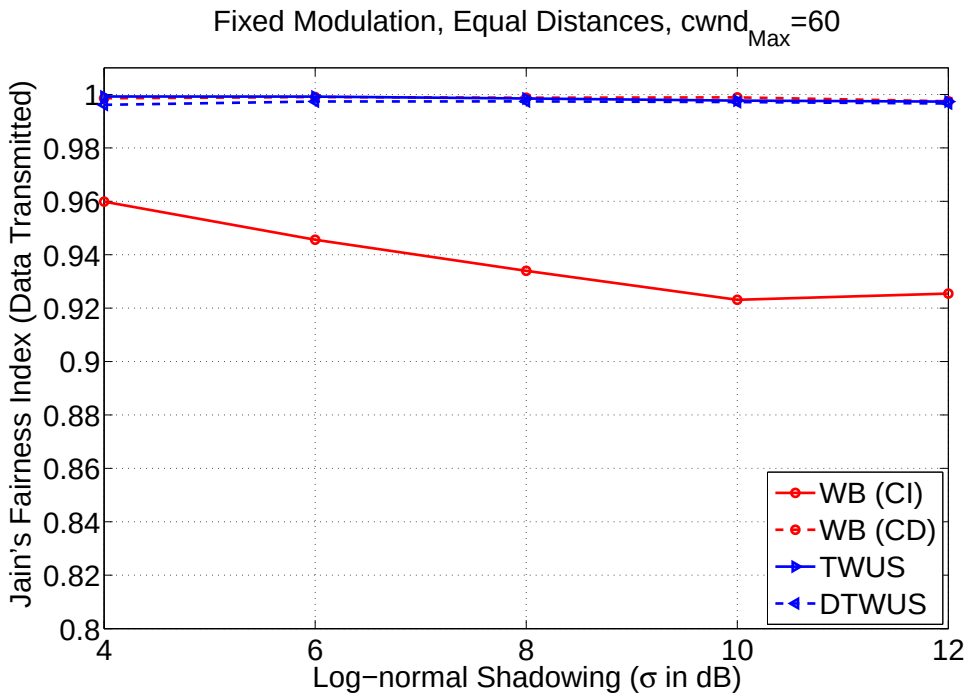


Figure 4.11: Jain's Fairness Index Comparison - Fixed Modulation, Equal Distances

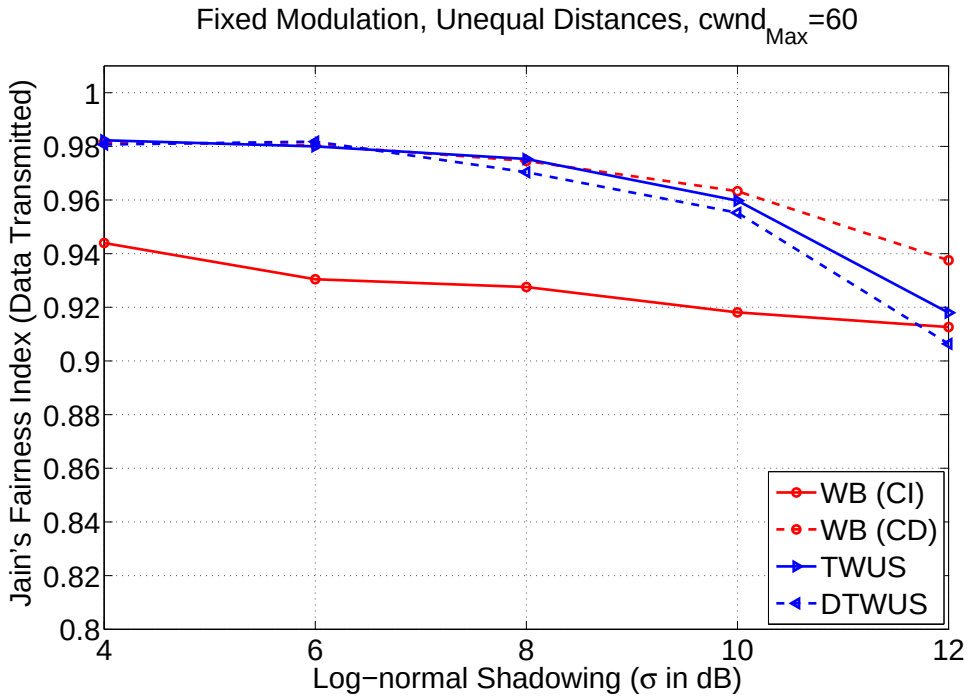


Figure 4.12: Jain's Fairness Index Comparison - Fixed Modulation, Unequal Distances

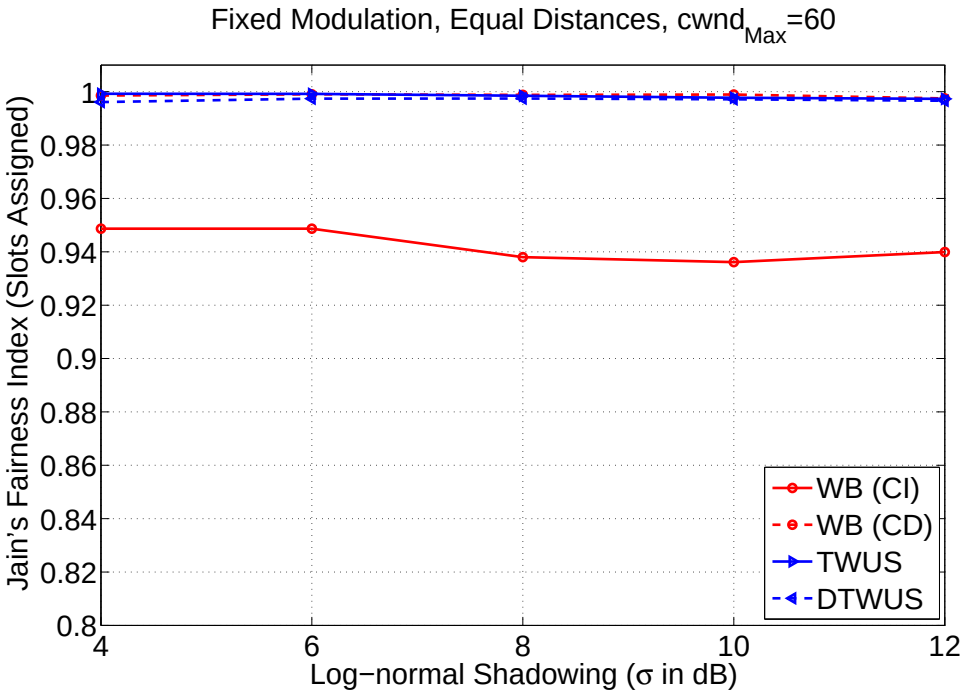


Figure 4.13: Jain's Fairness Index Comparison - Fixed Modulation, Equal Distances

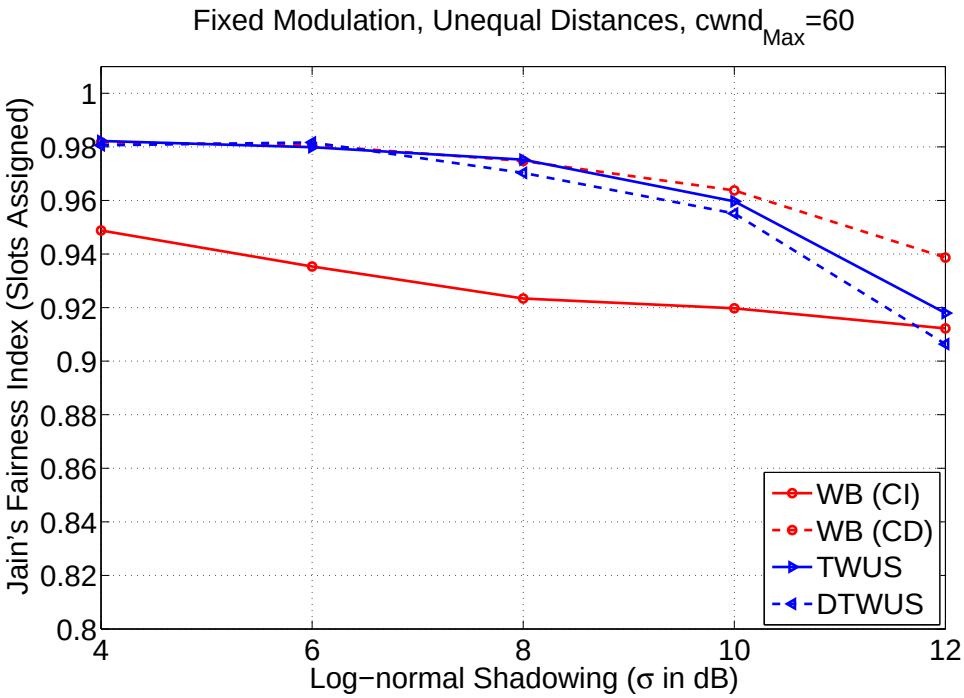


Figure 4.14: Jain's Fairness Index Comparison - Fixed Modulation, Unequal Distances

We also compare the fairness of the schedulers based on the number of slots assigned to different SSs under different shadowing. For this, we determine the JFI in number of slots assigned and repeat the same for both equal and unequal distance cases. We plot the JFI thus obtained in Figures 4.13 - 4.14. From these figures, we observe that JFI for the number of slots assigned remains above 90% in all experiments. These results show similar trends with that illustrated in Figures 4.11 - 4.12.

Slot Utilization and Idleness Comparison

We also investigate the slot utilization of the TCP-aware schedulers and compare it with WB schedulers. For this, we plot the percentage of slot utilization of TCP-aware as well as WB schedulers under different shadowing in Figures 4.15 - 4.16. From these figures, we observe that the slot utilization of TCP-aware schedulers is more than that of WB schedulers. We also observe that the utilization of TWUS scheduler is more than that of DTWUS scheduler and the utilization of WB (CD) scheduler is more than that of WB (CI) scheduler. For the equal distance experiment (Figure 4.15), slot utilization of TWUS and DTWUS scheduler varies between 84% to 90%, whereas that of WB (CD) varies between 60% to 75% and WB (CI) varies between 46% to 68%. We observe similar trends for the unequal distance experiments (Figure 4.16).

From Figures 4.15 - 4.16, we also observe that the slot utilization of WB (CI) schedulers falls as low as 40% when the Log-normal shadowing increases. This is because, when the channel is under heavy shadowing, the probability of getting polled is very low and the probability of not being scheduled is very high. This results in increase in delay in scheduling, which may triggers TCP timeout and drop in $cwnd$ size thereby resulting in low utilization. Utilization of slots can be further increased by adding different classes of traffic along with TCP.

Since the TCP-aware schedulers assign slots to a SS only if there is a requirement and the number of slots assigned is proportional to the resource requirement, the percentage of slot idleness is negligible, whereas it is not so in WB schedulers. We plot the slot idleness of WB schedulers at various Log-normal shadowing in Figures 4.17 - 4.18. From these figures, we observe that the slot idleness in WB (CI) scheduler is more than that of WB (CD) scheduler. This results in higher average $cwnd$ size and average TCP throughput for WB (CD) scheduler as compared to WB (CI) scheduler, explained in Figures 4.7 and 4.10.

We also compare the performance of TCP-aware schedulers with that of a RR scheduler, in the next section.

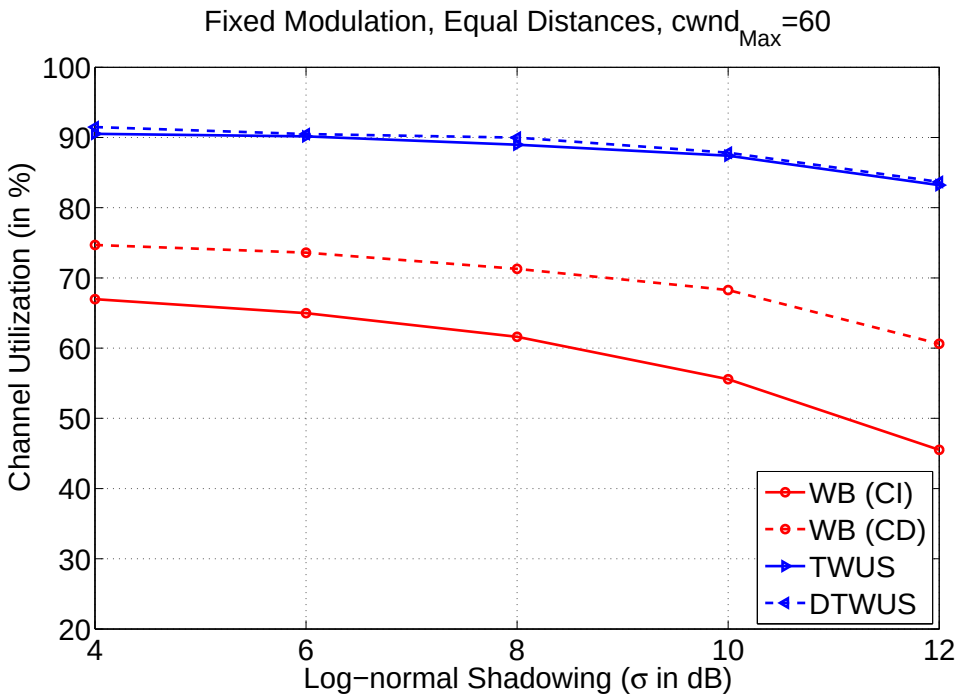


Figure 4.15: Channel Utilization - Fixed Modulation, Equal Distances

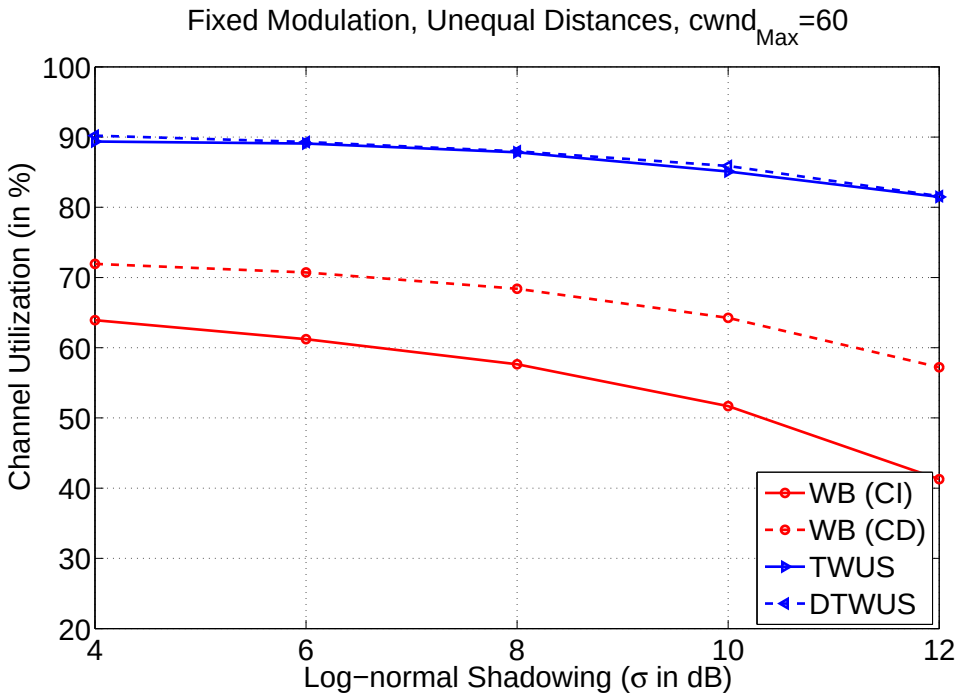


Figure 4.16: Channel Utilization - Fixed Modulation, Unequal Distances

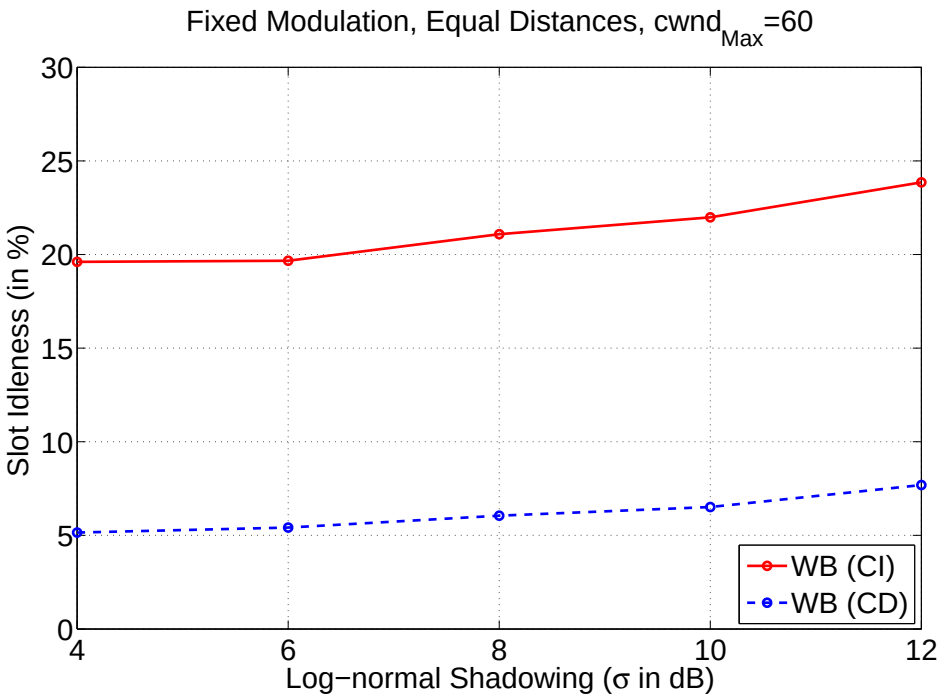


Figure 4.17: Slot Idleness - Fixed Modulation, Equal Distances

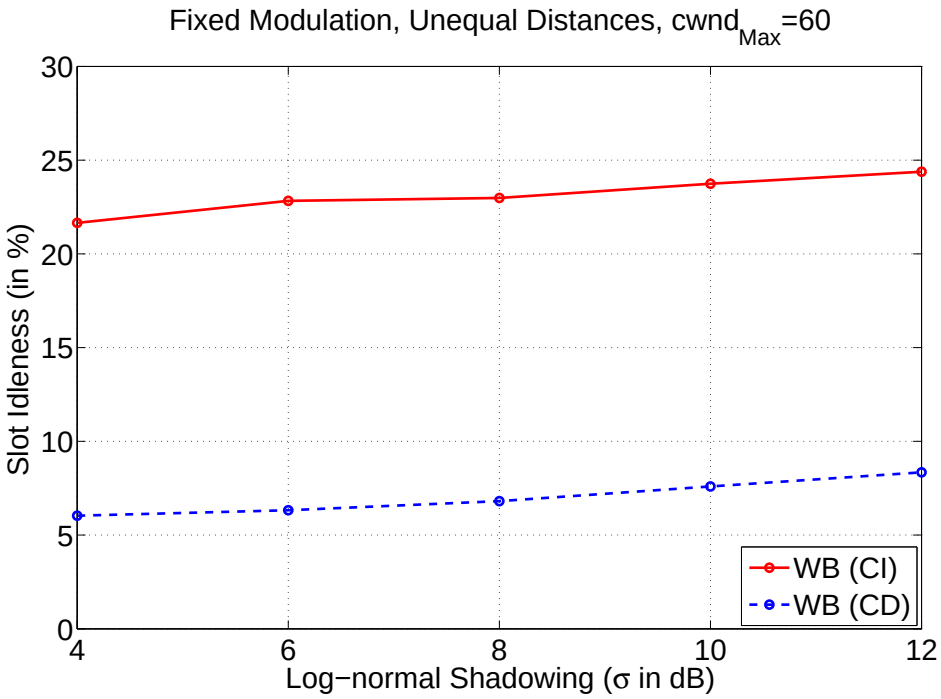


Figure 4.18: Slot Idleness - Fixed Modulation, Unequal Distances

4.4.3 Comparison with Round Robin Scheduler

We also implement a RR scheduler at the BS for comparison with TCP-aware schedulers. RR scheduler also determines the schedulable set at the beginning of the polling. The polling epoch k used in RR scheduler is same as that used in the TCP-aware schedulers. In RR scheduler, the BS schedules the active SSs in every frame in a round robin manner. We compare the average $cwnd$ size and average TCP throughput achieved by the RR as well as TCP-aware schedulers. In Figure 4.19, we plot the average $cwnd$ size under different shadowing with fixed modulation. From this figure, we observe that the average $cwnd$ size achieved by the TCP-aware schedulers are higher than that of RR schedulers for all shadowing cases. We also observe that as the σ of Log-normal shadowing increases, the average $cwnd$ size achieved by both RR and TCP-aware schedulers decreases. We also plot the average TCP throughput achieved by both RR and TCP-aware schedulers in Figure 4.20. From this figure, we observe that the average TCP throughput achieved by the TCP-aware schedulers is more as compared to that of RR schedulers. From Figure 4.19, we observe that both the average $cwnd$ size and TCP throughput achieved by DTWUS is more than that of TWUS. Though we have illustrated the results for equal distance experiments, similar results are also valid for unequal distance experiments.

We also combine the comparisons of the performance of TCP-aware schedulers with that of RR and WB schedulers. We plot the gain in $cwnd$ size and TCP throughput achieved in Figures 4.21 - 4.22. From these figures, we observe that the improvement of TCP-aware scheduler over WB schedulers is more as compared to that of the improvement over RR scheduler. Since DTWUS scheduler performs better than that of TWUS scheduler, we only plot the comparisons of TWUS scheduler with that of RR and WB schedulers. Though we have illustrated the results for equal distance experiments, similar results are also valid for unequal distance experiments.

4.5 Discussions

In this chapter, we have proposed TCP-aware uplink scheduling schemes with fixed modulation for a multipoint-to-point wireless network. We have performed extensive simulations and compare the performance of the TCP-aware schedulers with other popular wire-line schedulers such as Round Robin scheduler and Weight-based schedulers. In the next chapter, we incorporate adaptive modulation at the PHY layer and propose adaptive modulation based TCP-aware scheduling schemes.

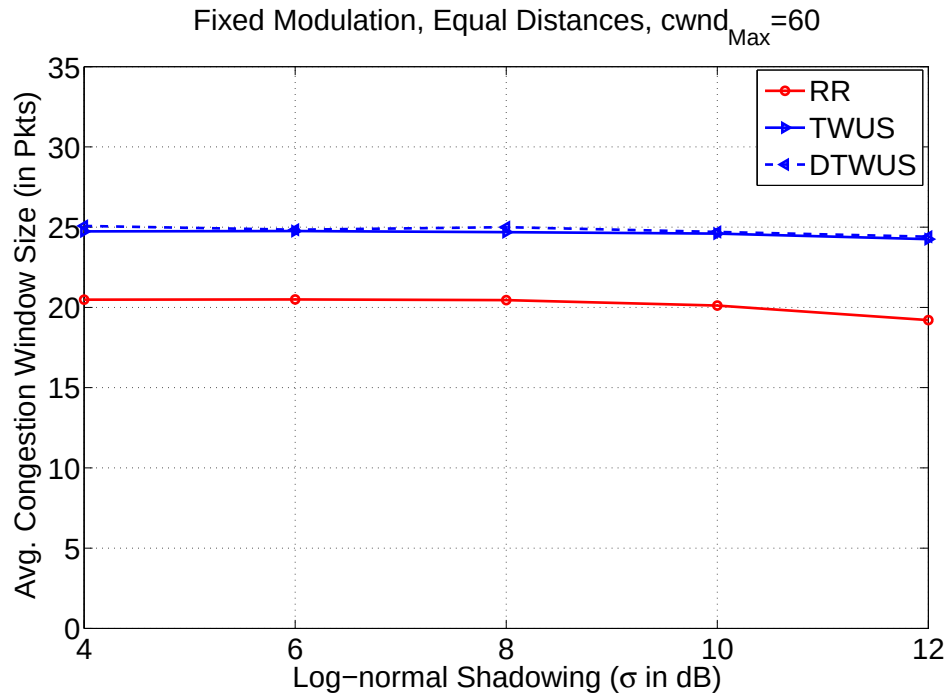


Figure 4.19: Comparison of TCP-aware Schedulers with RR Scheduler - Fixed Modulation, Equal Distances

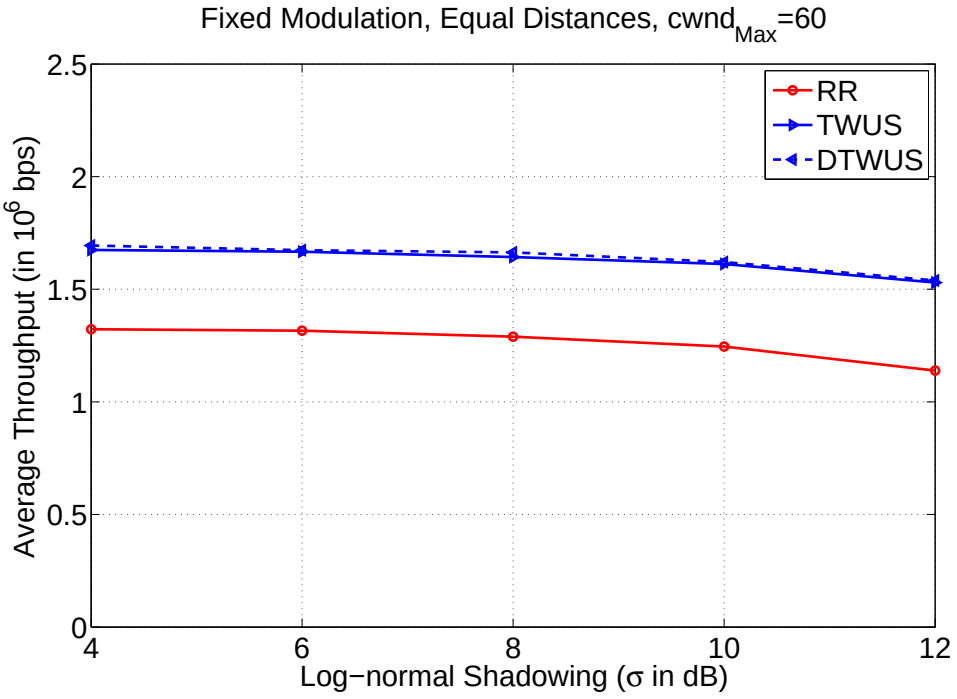


Figure 4.20: Comparison of TCP-aware Schedulers with RR Scheduler - Fixed Modulation, Equal Distances

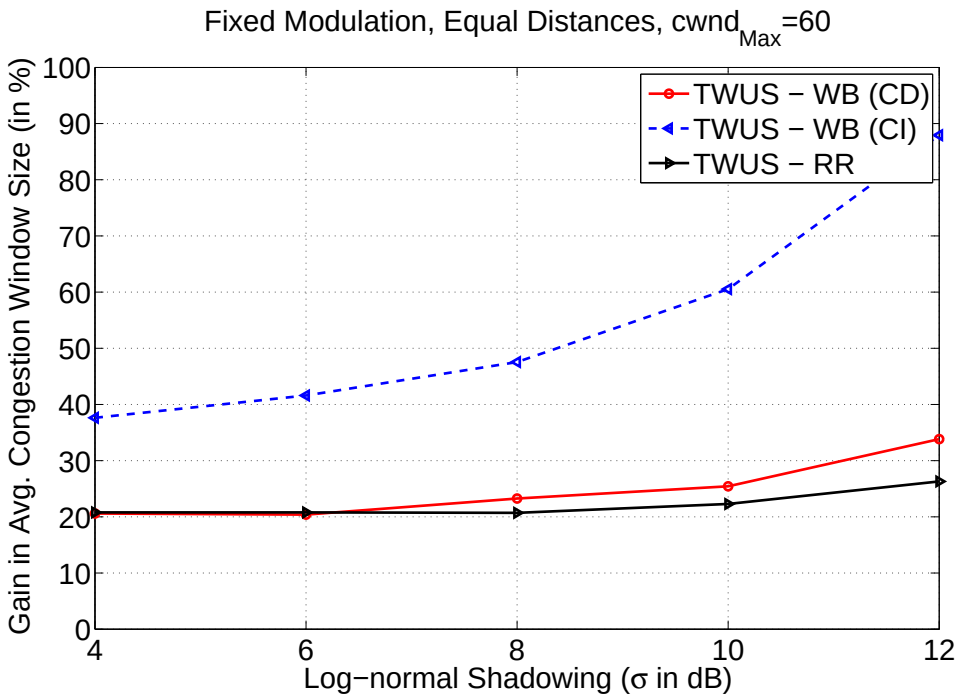


Figure 4.21: Gain in $cwnd$ size of TWUS over RR and WB Schedulers - Equal Distances

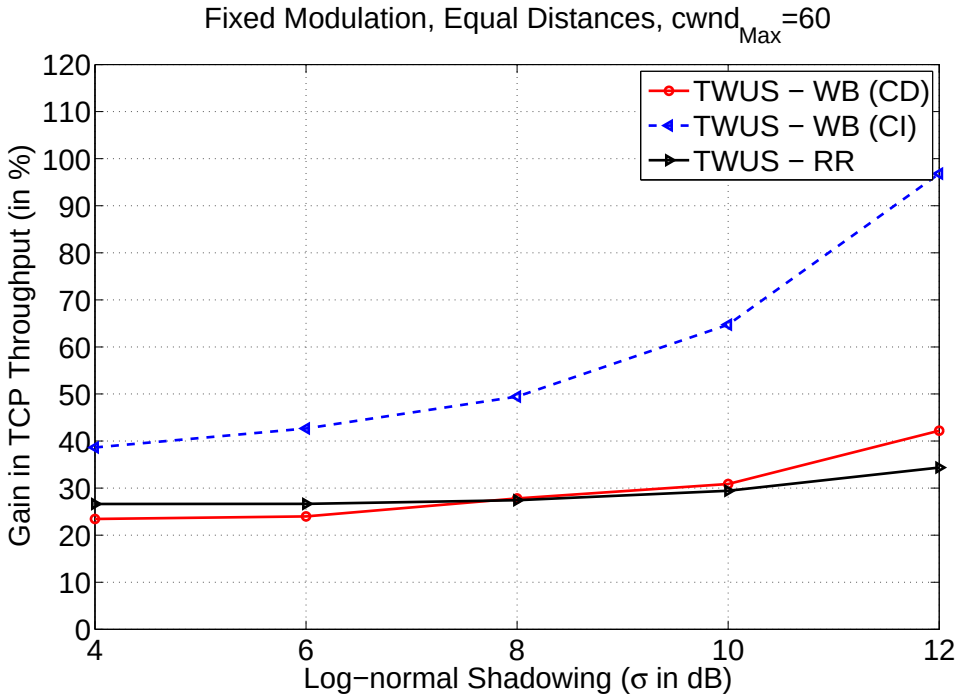


Figure 4.22: Gain in TCP Throughput of TWUS over RR and WB Schedulers - Equal Distances

Chapter 5

TCP-aware Fair Uplink Scheduling - Adaptive Modulation

In the previous chapter, we have proposed fixed modulation based TCP-aware fair uplink scheduling schemes in a multipoint-to-point wireless network. However, to achieve high spectral efficiency on fading channels, adaptive modulation and coding (AMC) can be used. Using AMC, a BS can choose a suitable modulation and coding scheme with the available SNR, such that the overall system capacity is increased. Therefore, we incorporate AMC in the proposed TCP-aware uplink scheduling algorithm. We also implement TCP-aware scheduling in the uplink of a multipoint-to-point IEEE 802.16 network and investigate its performance through exhaustive simulations. We compare the performance of TCP-aware uplink schedulers with adaptive modulation and fixed modulation. In addition, we compare the performance of TCP-aware schedulers with that of Weight-based (WB) and Round Robin (RR) schedulers taking adaptive modulation into consideration.

In Section 5.1, we discuss the system model of the TCP-aware schedulers. In Section 5.2, we present both TCP Window-aware Uplink Scheduling scheme and Deadline-based TCP Window-aware Uplink Scheduling scheme and illustrate the slot assignment methods. In Section 5.3, we discuss the implementation of TCP-aware schedulers in an IEEE 802.16 network. In Section 5.4, we discuss the experimental set up and present the results. These results demonstrate the efficiency of the proposed algorithm. In Section 5.5, we present TCP $send\ rate$ or average throughput determination of TCP-aware uplink schedulers in an IEEE 802.16 network and validate the analytical results with the simulation results. We discuss the fairness properties

of TCP-aware schedulers in Section 5.6.

5.1 System Model

The system model considered in this chapter is an extension of the system model that has been described in Section 4.2. As discussed in the previous chapter, we consider a multipoint-to-point wireless network, where multiple SSs are connected to one BS, as illustrated in Figure 5.1 (same as Figure 4.2). We incorporate adaptive modulation and coding at the PHY layer.

As discussed before, packets can be successfully received if $SNR_i \geq SNR_{th}$. The value of SNR_{th} depends upon the modulation and coding scheme used at the PHY layer. The maximum data rate attainable on the channel depends upon the modulation scheme and coding used.

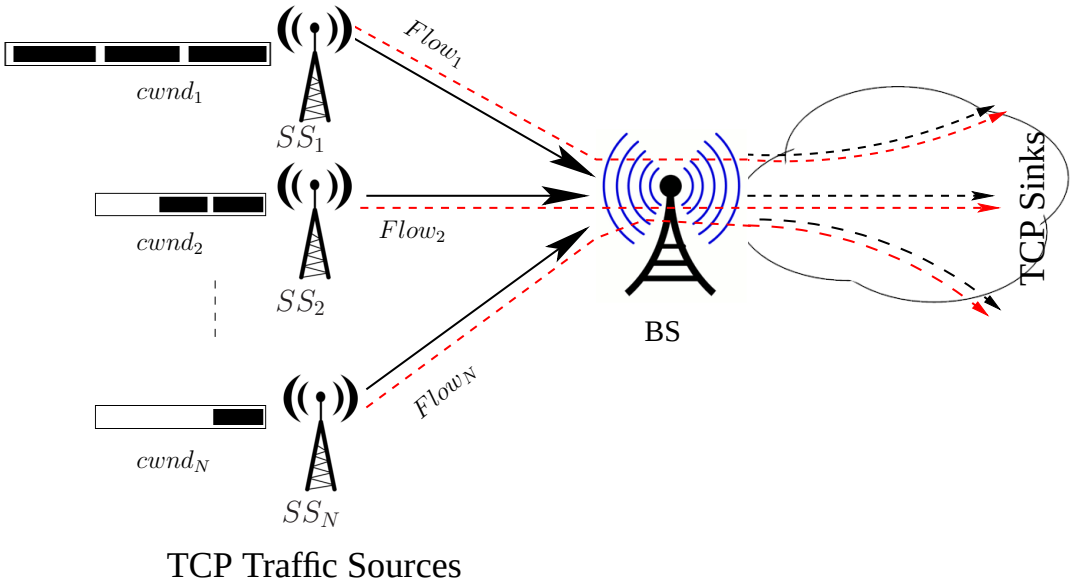


Figure 5.1: Multipoint-to-Point Framework with TCP-based Applications

5.2 Uplink Scheduling with Adaptive Modulation

Let R_i denote the rate of transmission between SS_i and the BS. Since wireless channel is time varying and the channel state changes from frame to frame, the modulation scheme suitable for that channel state also changes from frame to frame. Hence, $R_i(n)$ changes from frame to frame, with n being the frame index. Thus, defining quantum size in terms of number of slots assigned at the beginning of a polling epoch is not appropriate. We therefore, introduce $R_i(n)$ in the definition of the quantum size as follows:

$$
\begin{aligned}
Q(0) &= \frac{R_{min} N_s T_s}{M}, \\
Q(n) &= \frac{1}{M} \sum_{i \in L_{sch}} Flag_i(n-1) \times R_i(n-1) \times N_i(n-1) \times T_s, \forall i \in L_{sch},\ \forall n \geq 1,
\end{aligned}
\tag{5.1}
$$

where R_{min} is the minimum rate of transmission among all modulation schemes. From the above discussion, we observe that Q varies from frame to frame of a polling epoch when the PHY layer employs AMC, whereas Q is kept constant for a polling epoch when the PHY layer employs fixed modulation.

Similar to the classifications of the TCP-aware scheduler that has been proposed in the previous chapter, we classify the scheduling schemes into (i) TCP Window-aware Uplink Scheduling (TWUS-A) with Adaptive Modulation; and (ii) Deadline-based TCP Window-aware Uplink Scheduling (DTWUS-A) with Adaptive Modulation. We discuss the details of the scheduling algorithms in the subsequent sections.

5.2.1 Slot Allocation using TCP Window-aware Uplink Scheduling with Adaptive Modulation

This scheme is an extension of TWUS algorithm proposed in Section 4.3.2. The basic operation of it does not change significantly. Since the rate of transmission of different SSs are different, with the same number of slots, a SS with a higher transmission rate can transmit more number of packets as compared to another SS with a relatively lower transmission rate. Therefore, we introduce $R_i(n)$ into the resource requirement $D_i(n)$ and the deficit counter $DC_i(n)$. We update the resource requirement $D_i(n)$ of each schedulable SS in a frame n as follows:

$$
\begin{aligned}
D_i(0) &= cwnd_i \times PL, \\
D_i(n) &= cwnd_i \times PL - Tx_i(n-1) \\
&= D_i(n-1) - Flag_i(n-1) \times N_i(n-1) \times R_i(n-1) \times T_s,\ \ \forall n \geq 1.
\end{aligned}
\tag{5.2}
$$

We also update the deficit counter $DC_i(n)$ as follows:

$$
\begin{aligned}
DC_i(0) &= 1, \\
DC_i(n) &= DC_i(n-1) + Q(n) - Flag_i(n-1) \times R_i(n-1) \times N_i(n-1) \times T_s,\ \forall i \in L_{sch}, \\
&\qquad \forall n \geq 1.
\end{aligned}
\tag{5.3}
$$

As discussed before, the deficit counter can take a positive or negative value. Hence, we use scaled deficit counter $dc_i(n)$ for weight determination. The scaled deficit counter is updated in a manner similar to that of Eqn. (4.7). After determining the resource requirement and the scaled deficit counter, the BS determines the weights of each SS for assigning time slots. The weights are determined only for the active set members. For all other SSs, the weights are zero. The BS determines weights of active set SSs using the following equation:

$$W_i(n) = \frac{\frac{D_i(n)}{R_i(n)} \times \frac{dc_i(n)}{R_i(n)}}{\sum_{j \in L_{active}} \frac{D_j(n)}{R_j(n)} \times \frac{dc_j(n)}{R_j(n)}}, \forall i \in L_{active}, \ \ \forall n \geq 1. \tag{5.4}$$

Eqn. (5.4) essentially determines a weight $W_i(n)$ in frame n that is directly proportional to the normalized (with respect to rate $R_i(n)$) product of the scaled deficit counter and resource requirement. As discussed before, since, the rate of transmission is different for different SSs, with the same number of slots, a SS with a higher transmission rate can transmit more number of packets as compared to another SS with a relatively smaller transmission rate. Hence, allocating time slots in proportion only to $D_i(n)$ (or $cwnd_i$) may result in unfairness among the SSs. We use $R_i(n)$ in weight determination to ensure fairness among the SSs. After the determination of weights, the BS assigns slots to $SS_i, \forall i \in L_{active}$ in frame n using:

$$N_i(n) \ = \ \frac{1}{T_s} \times \min \left\{ \frac{W_i(n) \times T_f}{\sum_{j \in L_{active}} W_j(n)}, \frac{D_i(n)}{R_i(n)} \right\}, \forall i \in L_{active}, \ \ \forall n \geq 1. \tag{5.5}$$

The first term in the braces of Eqn. (5.5) corresponds to the number of slots in proportion to the weight $W_i(n)$, while the second term corresponds to the number of slots in proportion to the resource requirement $D_i(n)$ of SS_i. By using the $\min$ function in the above equation, the BS restricts the maximum number of slots assigned to any SS by its requirement. This ensures maximum slot utilization.

In the next section, we incorporate TCP timeout information during scheduling.

5.2.2 Slot Assignments using Deadline-based TCP Window-aware Uplink Scheduling with Adaptive Modulation

In this scheme, the method of determination of resource requirement, deficit counter and scaled deficit counter are similar to that defined in TWUS-A scheduler proposed in Section 5.2.1.

In addition to that we also use the TCP timeout information of the active flows during the scheduling process. To keep a track of the TCP timeout, we define deadline counter $d_i(n)$. The update deadline counter update follows that of Eqn. (4.10)[1].

After determining the scaled deficit counters as in Eqn. (4.7) and deadlines as in Eqn. (4.10), the BS determines the weight $W_i(n)$ of SS_i in frame n using the following equation:

$$W_i(n) = \frac{\frac{D_i(n)}{R_i(n)} \times \frac{dc_i(n)}{R_i(n)} / d_i(n)}{\sum_{j \in L_{active}} \frac{D_j(n)}{R_j(n)} \times \frac{dc_j(n)}{R_j(n)} / d_j(n)}, \ \forall i \in L_{active}, \ \ \forall n \geq 1. \tag{5.6}$$

Eqn. (5.6) is similar to Eqn. (5.4) except for the deadline $d_i(n)$. The use of the deadline in weight determination ensures that the weight of a SS that has a smaller deadline is higher as compared to that of another SS which has a larger deadline. After the determination of weights, the number of slots assigned to $SS_i, \forall i \in L_{active}$ in frame n is determined using Eqn. (5.5).

The pseudo-code of the proposed schedulers TWUS-A and DTWUS-A is presented in Algorithm 2. We have combined both schedulers by using $Flag_{deadline}$, which is set to one for DTWUS-A and is set to zero for TWUS-A.

5.3 Implementation of TCP-aware Scheduling

We consider TDD based IEEE 802.16 network, in which each frame of duration T_f is divided into uplink and downlink subframes of durations T_{ul} and T_{dl} respectively. We consider adaptive modulation scheme at the PHY layer. We employ QPSK, 16-QAM, 64-QAM modulation schemes as per the standard. Though the standard defines maximum baud rate, modulation schemes to be used and maximum data rate possible for WirelessMAN-SC interface, it does not specify the SNR thresholds for choosing different modulation schemes to be used.

The maximum data rate attainable for an Additive White Gaussian Noise (AWGN) channel can be expressed as: $R = B \times \log_2(1 + MI \times SNR)$, where R is the maximum attainable data rate and MI is the modulation index. Note that $MI = \frac{-\phi_1}{\log(\phi_2 BER)}$, where ϕ_1 and ϕ_2 are constants depending upon the modulation schemes [39]. Since the standard specifies data rates to be used, for a particular modulation scheme, SNR_{th} should satisfy:

[1]This scheme is also an extension of DTWUS scheme explained in Section 4.3.3.

$$
\begin{aligned}
SNR &= \frac{2^{\frac{R}{B}} - 1}{MI} \\
&= \frac{(1 - 2^{\frac{R}{B}}) \times \ln(5p_b)}{1.5}, \text{ if } \frac{R}{B} < 4 \\
&= \frac{(1 - 2^{\frac{R}{B}}) \times \ln(0.5p_b)}{1.5}, \text{ if } \frac{R}{B} \geq 4,
\end{aligned}
\tag{5.7}
$$

where p_b is the target bit error rate. Using Eqn. (5.7), we determine SNR_{th} for target BERs of 10^{-5} and 10^{-6} for a channel bandwidth (B) of 25 MHz. These are given in Table 5.1 for the data rate of 40, 80 and 120 Mbps for QPSK, 16-QAM and 64-QAM modulation schemes respectively.

Table 5.1: Modulation Schemes in the Uplink of WirelessMAN-SC IEEE 802.16 (Channel Bandwidth $B = 25$ MHz)

Modulation Scheme	**Data Rate** R **(Mbps)**	$\frac{R}{B}$ **(bps/Hz)**	SNR_{th} **(dB)** $BER = 10^{-5}$	SNR_{th} **(dB)** $BER = 10^{-6}$
QPSK	40	1.6	11.27	12.18
16-QAM	80	3.2	17.33	18.23
64-QAM	120	4.8	23.39	24.14

In the proposed scheme, SSs are required to maintain a queue (per flow) at their interfaces. Packets residing in the queue of a SS is served in a first-come first-serve basis. As per the standard, it is possible for the BS to determine the channel state of each SS. This information is used by the BS to determine the schedulable set at the beginning of polling and to update the active set in every frame. At the beginning of every polling epoch, each SS conveys its requirements in terms of its current congestion window ($cwnd_i$) size and time left to reach TCP timeout (TTO_i) to the BS. BS in turn, determines the number of slots to be assigned based on the resource requirement, deficit counter and deadline value of each schedulable SS and decides the modulation scheme to be used on a frame by frame basis. BS conveys this information to each SS through the uplink map. BS also determines the polling epoch k (as discussed in Section 4.3.1) such that the system efficiency and fairness are maintained. The block diagram of the proposed uplink scheduler is shown in Figure 5.2.

Algorithm 3 :TCP-aware Uplink Scheduler with Adaptive Modulation

while TRUE **do**
  Determine L_{sch} for the current polling epoch
  $Flag_i(0) \leftarrow 0 \; \forall i \in L_{sch}$
  $D_i(n) \leftarrow cwnd_i \times PL, \; DC_i(0) \leftarrow 1, \; dc_i(0) \leftarrow 1, \; W_i(0) \leftarrow 0, \; N_i(0) \leftarrow 0 \; \forall i \in L_{sch}$
  if $Shceduler_{Type} = TWUS - A$ **then**
    $Flag_{deadline} = 0, \;\; d_i(0) \leftarrow 1 \; \forall i \in L_{sch}$
  else
    $Flag_{deadline} = 1, \;\; d_i(0) \leftarrow TTO_i \; \forall i \in L_{sch}$
  end if
  $M \leftarrow |L_{sch}|$
  if $n = 1$ **then**
    $Q(0) \leftarrow \frac{R_{min} \times N_s \times T_s}{M}$
  end if
  $k \leftarrow \min_i \{RTT_i\}, \; T \leftarrow kT_f$
  Frame number $n \leftarrow 1$
  while $T > 0$ **do**
    $L_{active} \leftarrow \phi$
    for all $i \in L_{sch}$ **do**
      if $(SNR_i(n) \geq SNR_{th}) \; \Lambda (D_i(n-1) > 1)$ **then**
        $L_{active} \leftarrow L_{active} \cup \{i\}$
        $DC_i(n) \leftarrow DC_i(n-1) + Q(n-1) - R_i(n-1) \times N_i(n-1) \times T_s$
        if $Flag_{deadline}$ = 1 **then**
          $d_i(n) \leftarrow d_i(n-1)$
        else
          $d_i(n) \leftarrow 1$
        end if
      else
        $R_i(n) \leftarrow 0, \; D_i(n) \leftarrow D_i(n-1), \; DC_i(n) \leftarrow DC_i(n-1) + Q(n-1)$
        if $Flag_{deadline}$ = 1 **then**
          $d_i(n) \leftarrow d_i(n-1) - T_f$
        else
          $d_i(n) \leftarrow 1$
        end if
        if $d_i(n) \leq 0$ **then**
          $d_i(n) \leftarrow TO_i$
        end if
        $W_i(n) \leftarrow 0, \; N_i(n) \leftarrow 0$
      end if
    end for
    for all $i \in L_{active}$ **do**
      $D_i(n) \leftarrow D_i(n-1) - N_i(n-1) \times R_i(n-1) \times T_s$
      $dc_i(n) \leftarrow DC_i(n) + \min_j |DC_j(n)|, \forall j \in L_{active}$
      Map $R_i(n)$ to $SNR_i(n)$ in Table 5.1
      $W_i(n) \leftarrow \frac{\frac{D_i(n)}{R_i(n)} \times \frac{dc_i(n)}{R_i(n)} / d_i(n)}{\sum_{j \in L_{active}} (\frac{D_j(n)}{R_j(n)} \times \frac{dc_j(n)}{R_j(n)} / d_j(n))}$
      $N_i(n) \leftarrow \frac{1}{T_s} \times \min \left(\frac{W_i(n) \times T_f}{\sum_{j \in L_{active}} W_j(n)}, \frac{D_i(n)}{R_i(n)} \right)$
      $Q(n) \leftarrow \frac{1}{M} \sum_{i \in L_{sch}} R_i(n-1) \times N_i(n-1) \times T_s$
    end for
    $T \leftarrow T - T_f, \; n \leftarrow n + 1$
  end while
end while

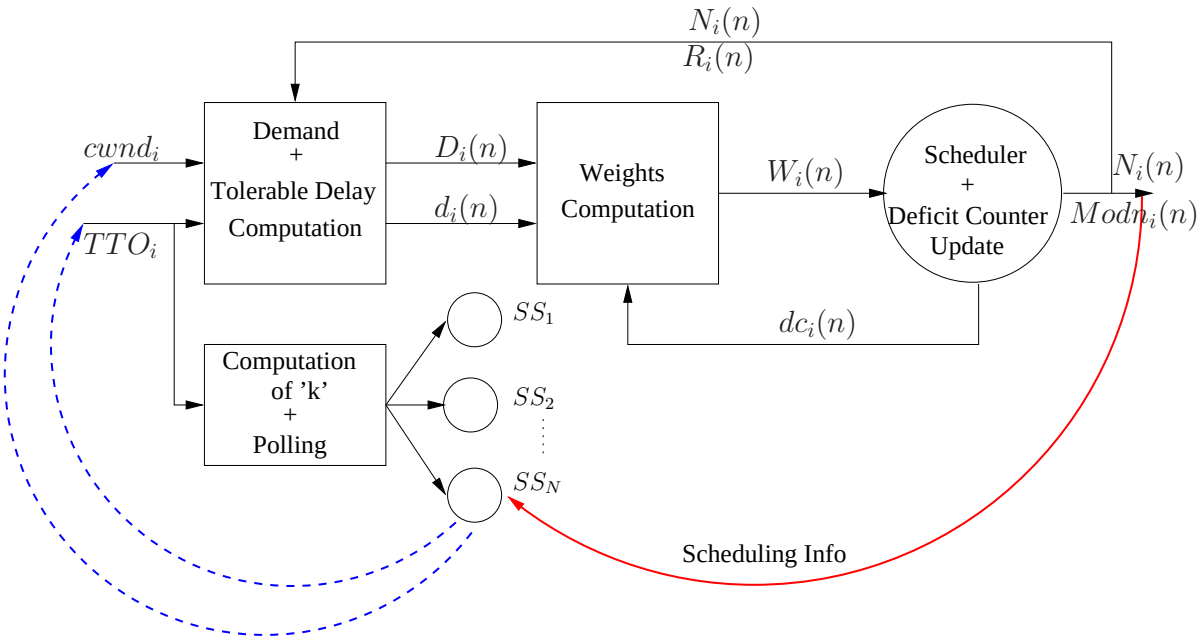


Figure 5.2: Block Diagram of TCP-aware Uplink Scheduler

5.4 Experimental Evaluation of TCP-aware Schedulers with Adaptive Modulation

In this section, we describe simulation experiments that have been performed to evaluate TCP-aware scheduling with adaptive modulation. All the simulations have been conducted using implementations of TCP-aware scheduling with IEEE 802.16 setting in MATLAB [68]. We consider a multipoint-to-point IEEE 802.16 based network where 10 SSs are connected to a centralized BS as shown in Figure 5.1. We consider $BER = 10^{-6}$ for the applications and use the SNR_{th}s for selecting an appropriate modulation scheme as shown in Table 5.1. Except for the addition of adaptive modulation and coding at the PHY layer, the experimental setup used here is identical to that of Section 4.4. The system parameters used for simulations are presented in Table 5.2. The value of each parameter observed has been averaged over 50 independent simulation runs.

We conduct four sets of experiments based on distance (equal and unequal), and the proposed schedulers (TWUS-A and DTWUS-A). For each set of experiment, we vary the standard deviation (σ) of Log-normal shadowing between 4, 6, 8, 10 and 12 dB.

Table 5.2: Summary of System Parameters

Simulation Parameter	**Value**
Channel Bandwidth	25 MHz
Adaptive Modulation Schemes	QPSK, 16-QAM, 64-QAM
Bit Error Rate	10^{-6}
Path Loss Exponent (γ)	4
Frame Length T_f	2 msec
Uplink/Downlink Frame Length	1 msec
Number of Data Slots per T_{ul}	500
Number of Frames Simulated	40000
TCP Type	TCP Reno
Number of Independent Runs	20
No. of SSs	10
Packet Size	8000 bits

5.4.1 Simulation Results

Impact of $cwnd_{Max}$

We perform experiments to determine the value of $cwnd_{Max}$ at which the TCP throughput saturates and plot the results in Figure 5.3. For completeness, we plot the results of both TWUS and TWUS-A with equal distances in this figure. From this figure, we observe that TCP throughput remains constant (reaches saturation) once the $cwnd_{Max}$ reaches 70 packets for TWUS-A, whereas it is 60 packets for TWUS. We choose $cwnd_{Max}$ = 70 for the TCP-aware schedulers with adaptive modulation scheme in the rest of our experiments.

5.4.2 Comparison With Weight-based Schedulers

We also compare the performance of TCP-aware schedulers with that of WB (CI) and WB (CD) schedulers taking adaptive modulation into consideration. We determine the resource requirement of each user using Eqn. (5.2). Similar to the fixed modulation experiments, we determine the average $cwnd$ size, average TCP throughput, slot utilization and Jain's Fairness Index (JFI) achieved by each of the schedulers for a fair comparison.

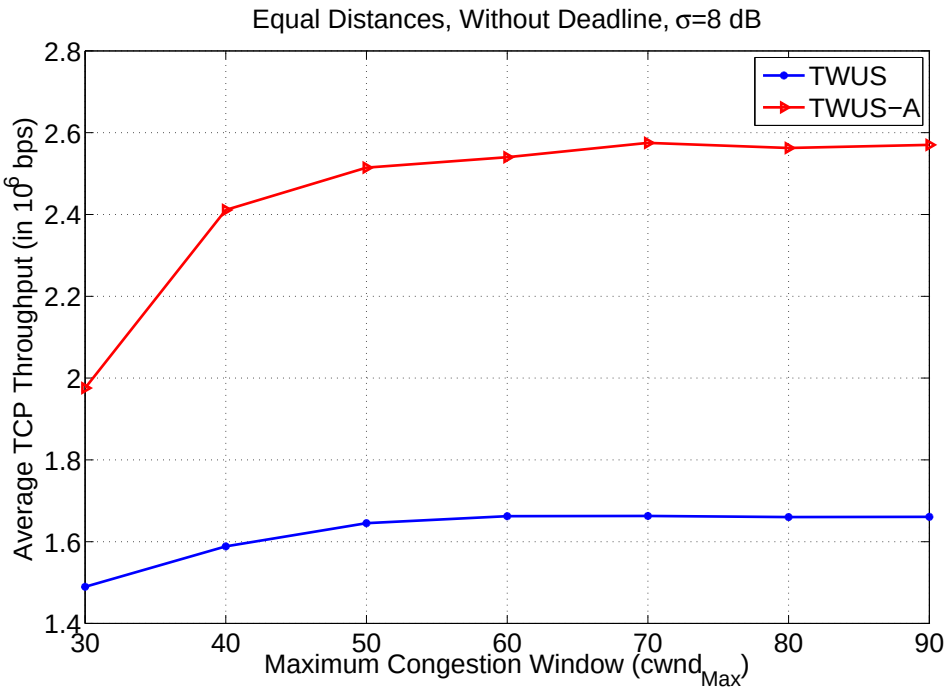


Figure 5.3: Average TCP Throughput vs. $cwnd_{Max}$

Average $cwnd$ size Comparison

In Figures 5.4 - 5.5, we plot the average $cwnd$ size achieved by the WB schedulers and TCP-aware schedulers under different standard deviation (σ) of Log-normal shadowing with adaptive modulation. The distance between the SSs and the BS are considered to be equal in Figure 5.4 and unequal in Figure 5.5. From these figures, we observe that the average $cwnd$ size achieved by the TCP-aware schedulers is higher than that of WB schedulers for all shadowing cases. We also observe that as standard deviation of Log-normal shadowing increases, the average $cwnd$ size achieved by both WB and TCP-aware schedulers decreases. However, we observe that the rate of decrease of $cwnd$ is more in adaptive modulation (Figures 5.4 - 5.5) than that in fixed modulation (Figures 4.5 - 4.6).

In Figure 5.6, we plot the percentage increase in the average $cwnd$ size by the TCP-aware schedulers over WB schedulers under different Log-normal shadowing. From this figure, we observe that as the standard deviation (σ) of the Log-normal shadowing increases, the gain in $cwnd$ size also increases. Moreover, the gain in $cwnd$ size of TWUS-A over WB (CI) varies between 22% to 50%, whereas it varies between 38% to 91% for TWUS-A over WB (CD). Though we have illustrated the results for equal distance experiments, similar comparisons are also valid for unequal distance experiments.

We also compare the performance of WB (CD) and WB (CI) schedulers with adaptive modulations. From Figures 5.4 - 5.5, we observe that the average $cwnd$ size achieved by the WB (CD) scheduler is more than that of WB (CI) scheduler, which is similar to that of the fixed modulation (Figures 4.5 - 4.6). However, the difference in average $cwnd$ size achieved by the WB (CD) scheduler and WB (CI) scheduler with adaptive modulation is higher than that achieved with fixed modulation. Since scheduling in WB (CD) is channel dependent, it is more adaptive than WB (CI) to the variable rates than the fixed rate.

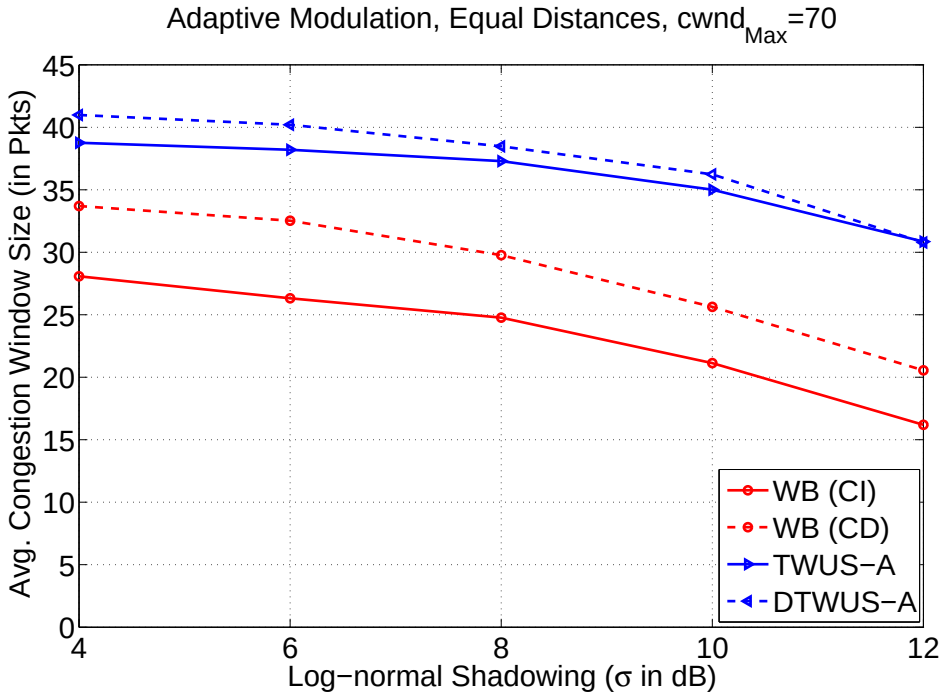


Figure 5.4: Average $cwnd$ size Comparison - Adaptive Modulation, Equal Distances

Average Throughput Comparison

We also compare the average TCP throughput achieved by the schedulers with adaptive modulation schemes. In Figures 5.7 - 5.8, we plot the average TCP throughput obtained by the TCP-aware schedulers and WB schedulers under different shadowing and distances. We plot the gain in average TCP throughput achieved by the TCP-aware scheduler over the WB schedulers in Figure 5.9. The improvements observed in TCP throughput is similar to that of the improvement in average $cwnd$ size illustrated in Figure 5.6. We also observe that the gain in average TCP throughput of TWUS-A over WB (CI) varies between 25% to 60%, whereas it varies between 38% to 100% for TWUS-A over WB (CD).

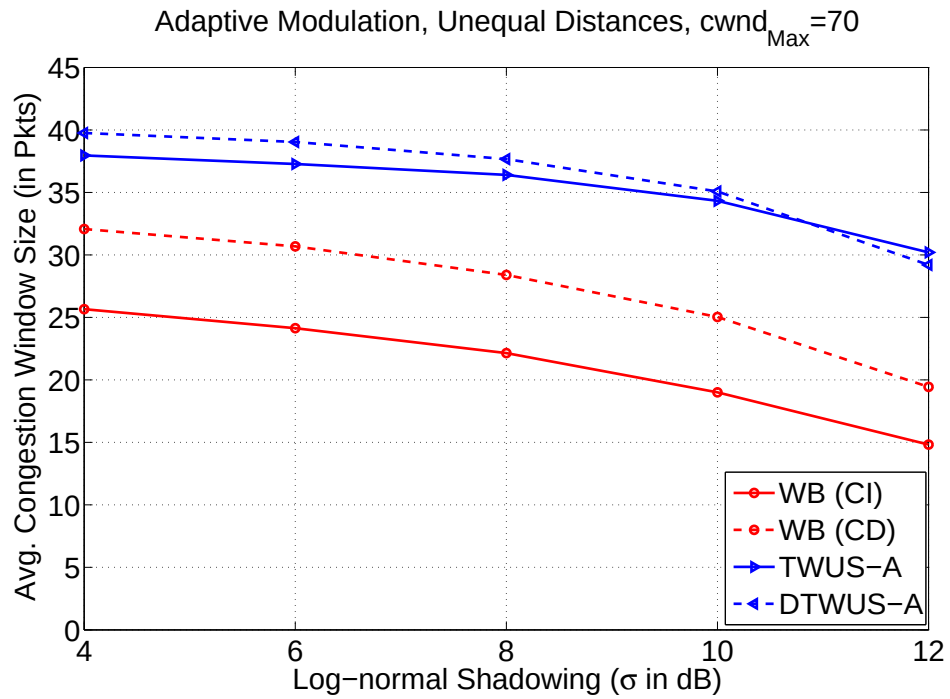


Figure 5.5: Average $cwnd$ size Comparison - Adaptive Modulation, Unequal Distances

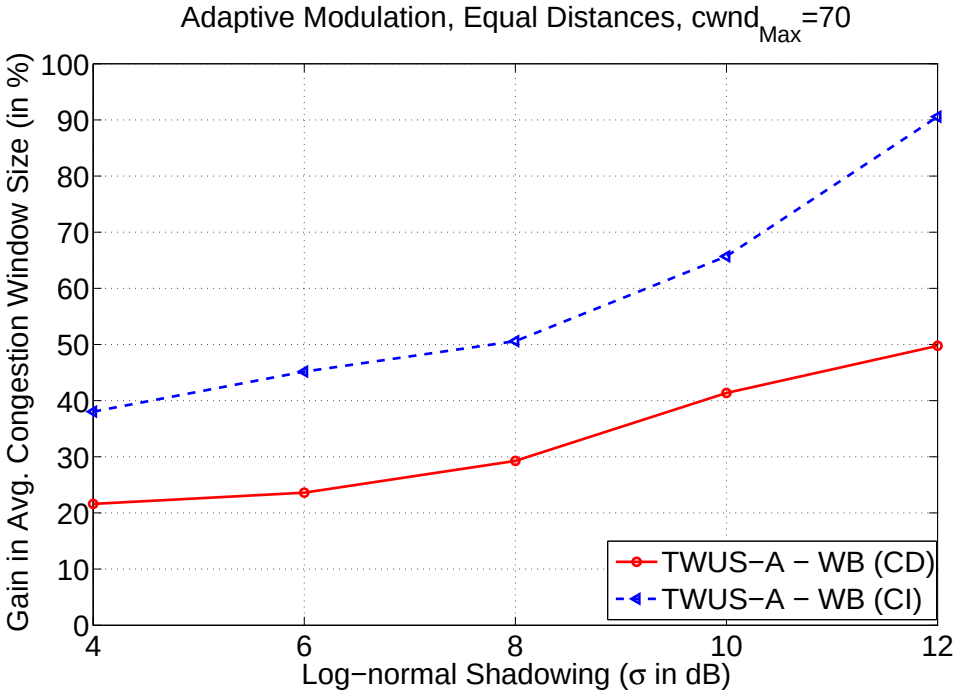


Figure 5.6: Gain in $cwnd$ size of TWUS-A over WB Schedulers - Equal Distances

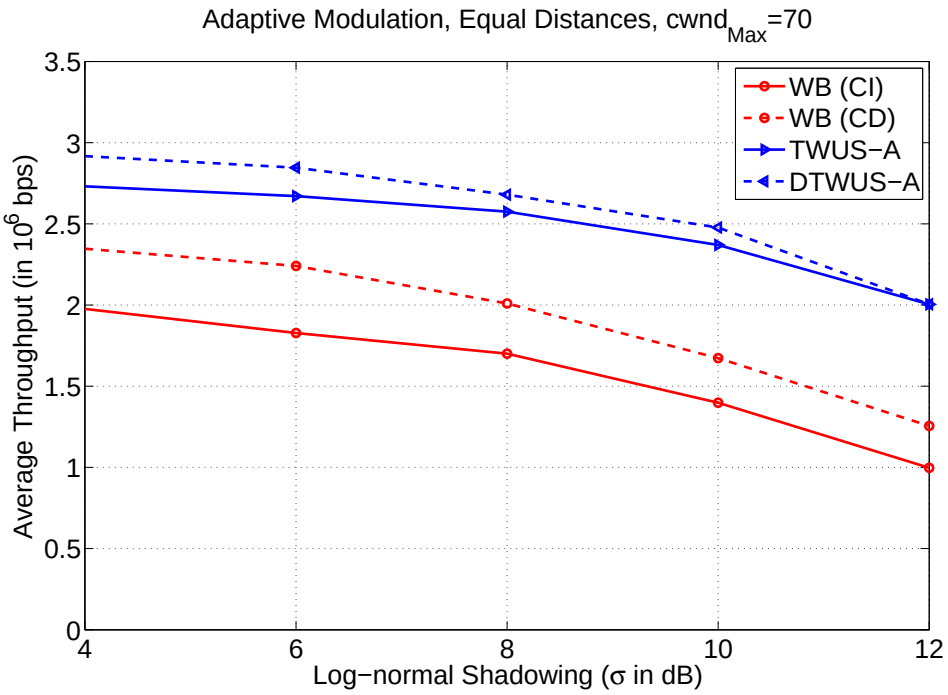


Figure 5.7: TCP Throughput Comparison - Adaptive Modulation, Equal Distances

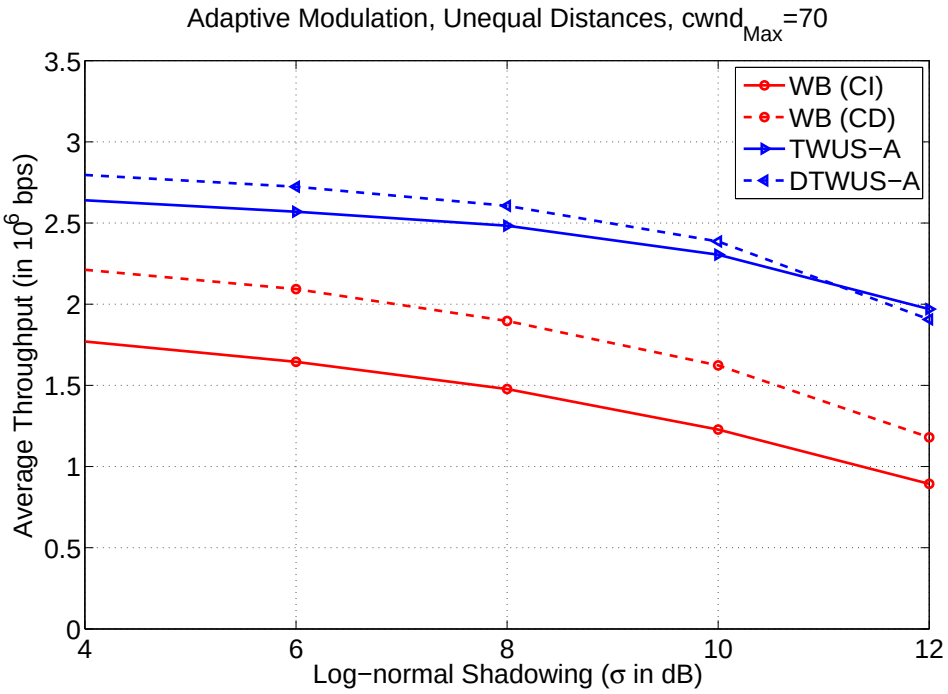


Figure 5.8: TCP Throughput Comparison - Adaptive Modulation, Unequal Distances

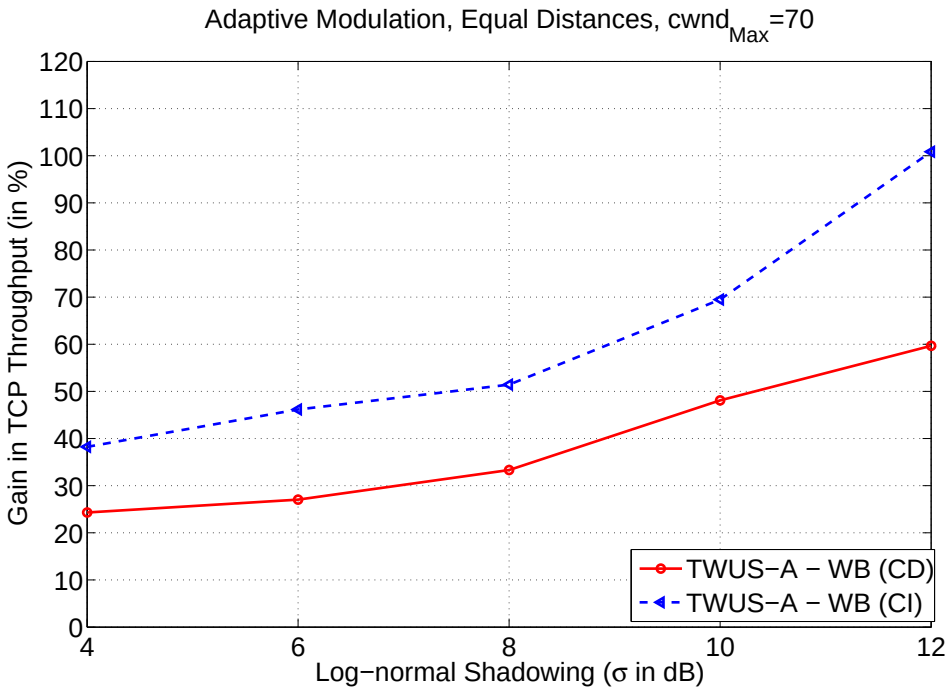


Figure 5.9: Gain in TCP Throughput of TWUS-A over WB Schedulers - Equal Distances

Fairness Comparison

We also determine the JFI for the amount of data transmitted by each SS and the number of slots assigned to each SS for the TCP-aware and WB schedulers with adaptive modulation. In Figures 5.10 and 5.11, we plot the variation of JFI thus determined for different value of σ of Log-normal shadowing. From these figures, we observe that both TWUS-A and DTWUS-A are fair under various shadowing conditions. Moreover, JFI obtained by TWUS-A and DTWUS-A schedulers are higher than that of WB (CI) scheduler. However, JFI obtained by TWUS-A and DTWUS-A schedulers are in the same range with that of WB (CD) scheduler. This is due to the fact that WB (CD) is adaptive to the channel states and attempts to schedule the users in such a manner that the amount of data transmitted by each user is the same. We also observe that unlike fixed modulation, the JFI of TWUS-A and DTWUS-A decreases as standard deviation of Log-normal shadowing increases. Though we have illustrated the results for equal distance experiments, similar results are also valid for unequal distance experiments.

Slot Utilization and Idleness Comparison

We investigate the slot utilization of TWUS-A and DTWUS-A and compare it with WB (CD) and WB (CI) schedulers. For this, we plot the percentage in slot utilization under different

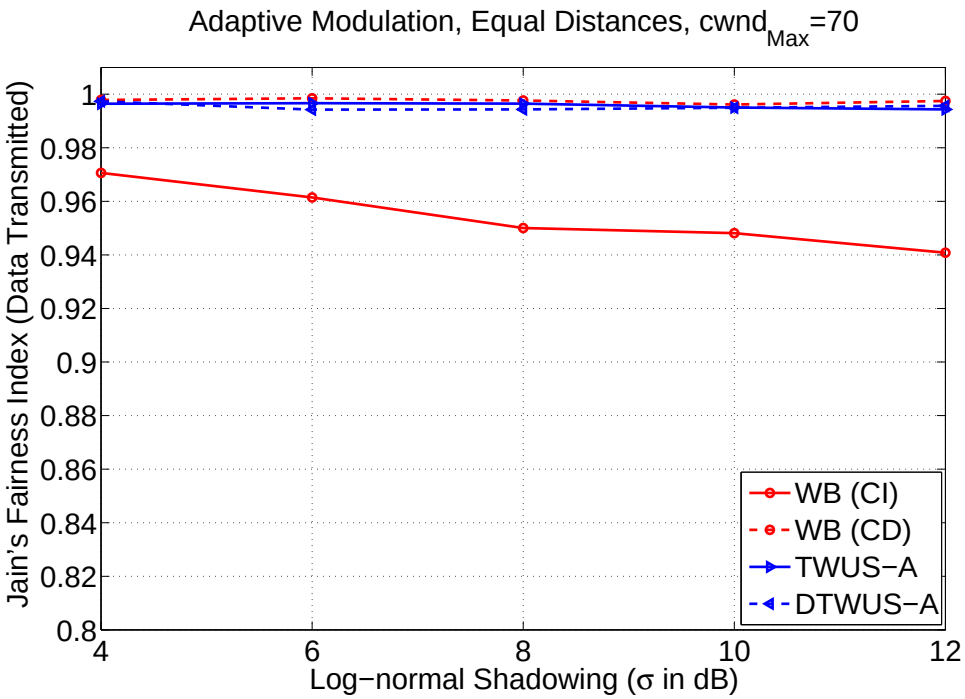


Figure 5.10: Jain's Fairness Index Comparison - Adaptive Modulation, Equal Distances

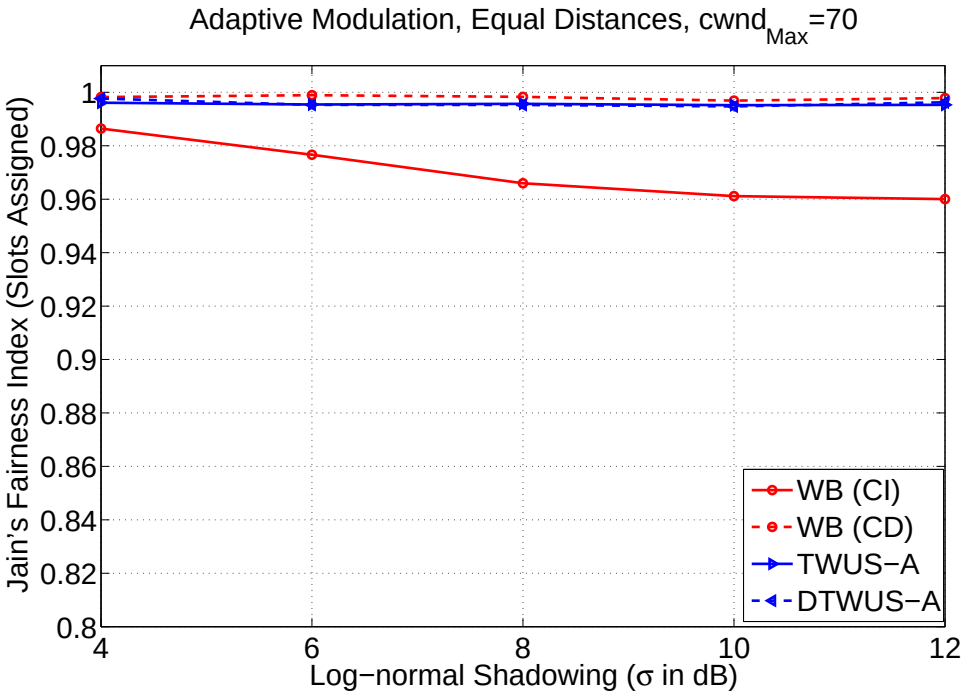


Figure 5.11: Jain's Fairness Index Comparison - Adaptive Modulation, Equal Distances

shadowing and distances in Figures 5.12 - 5.13. From these figures, we observe that the slot utilization of TCP-aware schedulers is more than that of WB schedulers. We also observe that the utilization of TWUS-A scheduler is more than that of DTWUS-A scheduler and the utilization of WB (CD) scheduler is more than that of WB (CI) scheduler. Moreover, the slot utilization of TWUS-A and DTWUS-A scheduler varies between 70% to 85%, whereas that of WB (CI) and WB (CD) varies between 40% to 60%. Similar to the results obtained in the fixed modulation experiments, in adaptive modulation, we also observe that the slot utilization of WB (CI) scheduler falls as low as 40% when the standard deviation of Log-normal shadowing increases. This is because, when the channel is under heavy shadowing, the probability of getting polled is very low and the probability of not being scheduled in a frame is very high. This results in congestion and drop in $cwnd$ size thereby resulting in low utilization. Utilization of slots can be further increased by adding different classes of traffic along with TCP traffic.

We also compare the slot idleness of WB (CD) and WB (CI) schedulers with adaptive modulation. We plot the slot idleness of WB schedulers under different σ of Log-normal shadowing in Figures 5.14 - 5.15. From these figures, we observe that the slot idleness in WB (CI) scheduler is more than that of WB (CD) scheduler. This results in higher average $cwnd$ size and average TCP throughput for WB (CD) scheduler as compared to WB (CI) scheduler, observed in Figures 5.6 - 5.9.

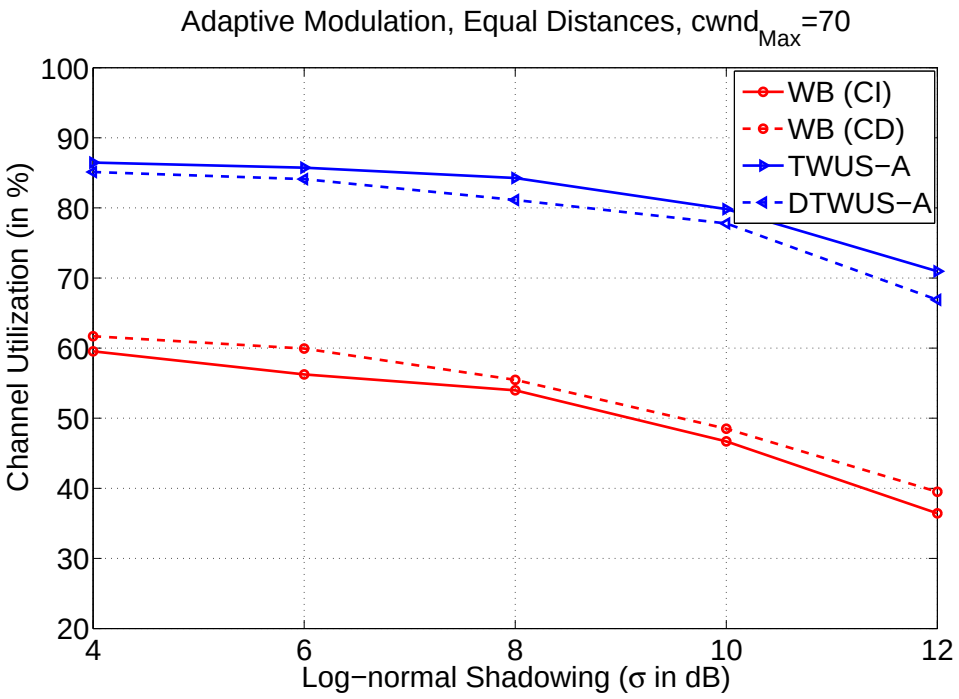


Figure 5.12: Channel Utilization - Adaptive Modulation, Equal Distances

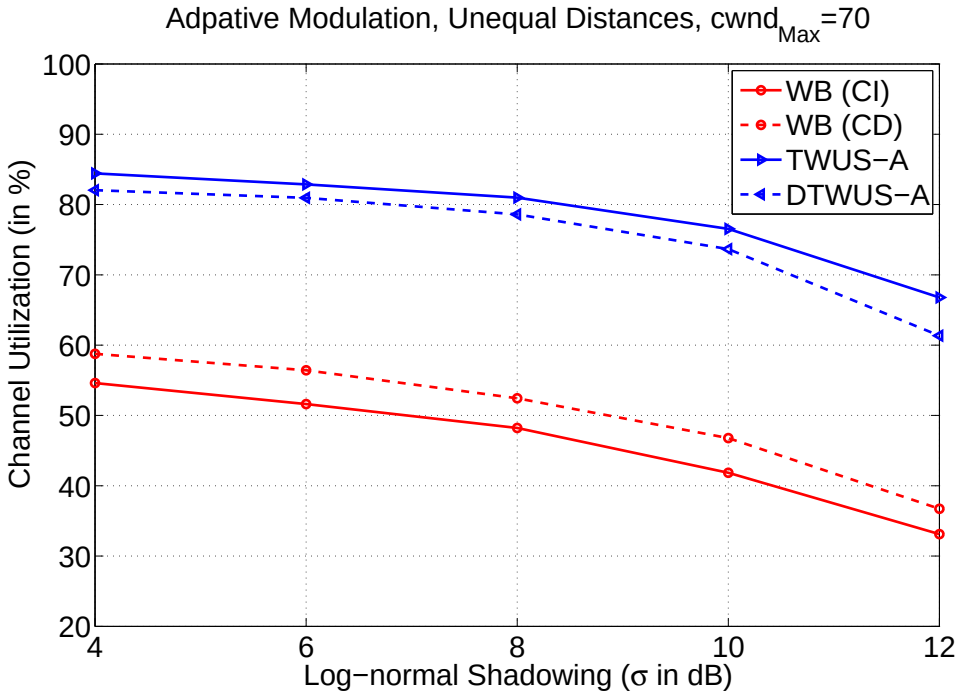


Figure 5.13: Channel Utilization - Adaptive Modulation, Unequal Distances

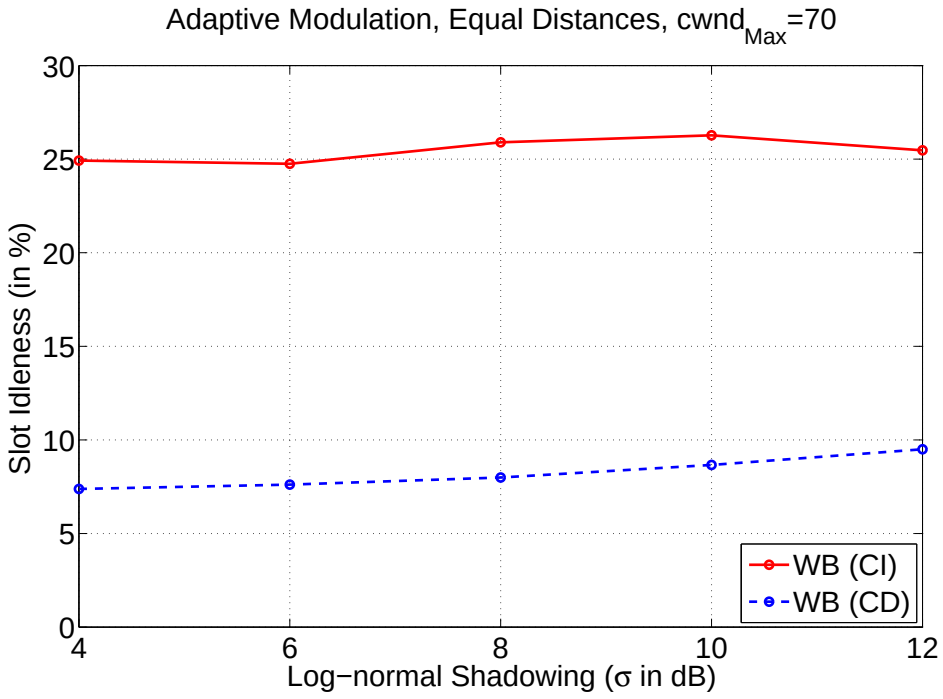


Figure 5.14: Slot Idleness - Adaptive Modulation, Equal Distances

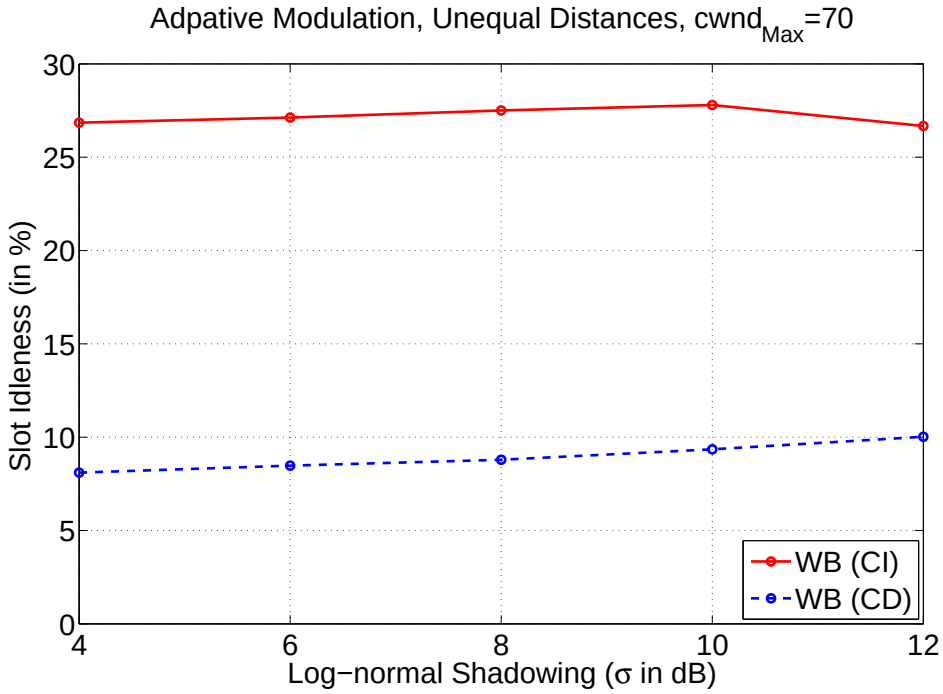


Figure 5.15: Slot Idleness - Adaptive Modulation, Unequal Distances

We also compare the performance of TCP-aware schedulers with that of a RR scheduler with adaptive modulation in the next section.

5.4.3 Comparison with Round Robin Scheduler

We compare the performance of TCP-aware schedulers with that of a Round Robin (RR) scheduler taking adaptive modulation into consideration. Similar to the fixed modulation experiment, we compare the average $cwnd$ size and average TCP throughput achieved by the RR as well as TCP-aware schedulers. In Figures 5.16 and 5.17, we plot the average $cwnd$ size and TCP throughput obtained respectively under different shadowing with adaptive modulation in consideration. From these figures, we observe that the average $cwnd$ size as well as TCP thoughput achieved by the TCP-aware schedulers are higher than that of RR schedulers for all Log-normal shadowing cases. Similar to the results obtained with fixed modulation, we also observe that with adaptive modulation, as the σ of Log-normal shadowing increases, the average $cwnd$ size as well as TCP throughput achieved by both RR and TCP-aware schedulers decreases. However, the rate of decrease of $cwnd$ and TCP throughput is more in adaptive modulation than that in fixed modulation (Figures 4.19- 4.20). Though, we have illustrated the results for equal distance experiments, similar results are also valid for unequal distance experiments.

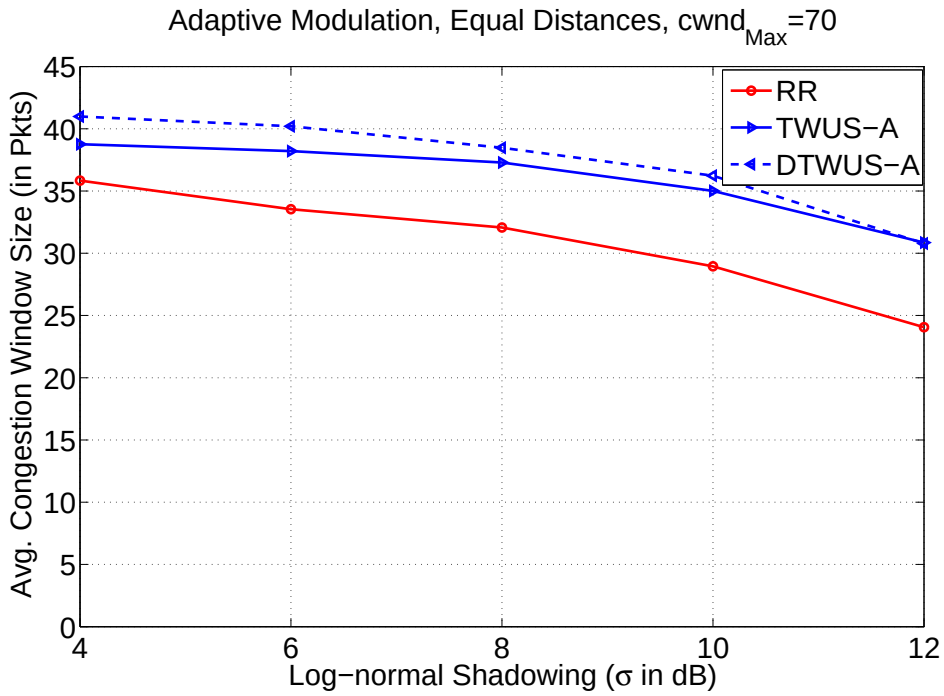


Figure 5.16: Comparison of TCP-aware Schedulers with RR Scheduler - Adaptive Modulation, Equal Distances

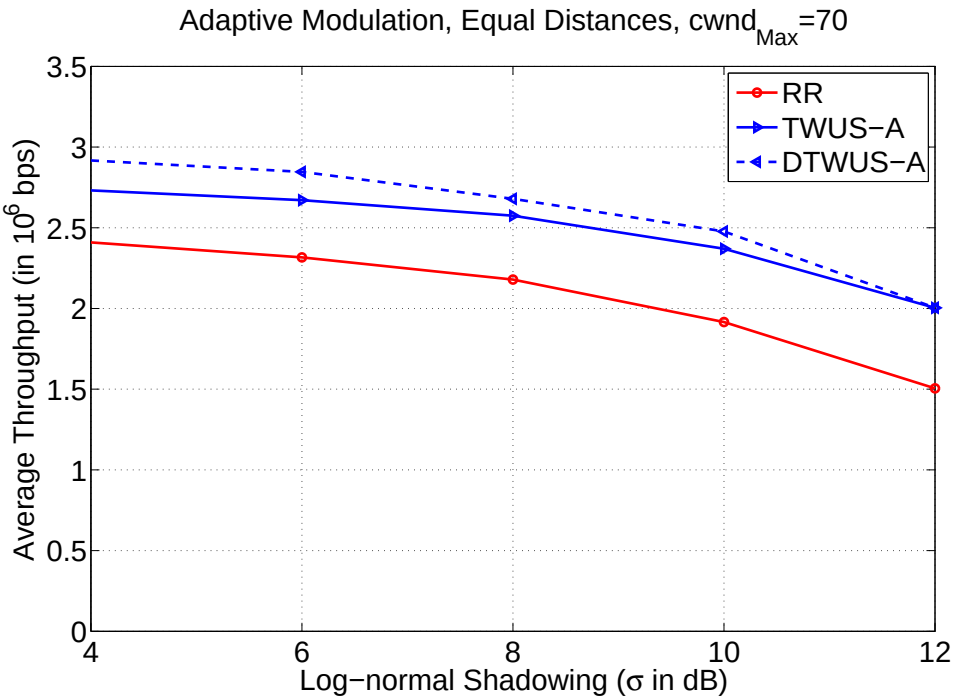


Figure 5.17: Comparison of TCP-aware Schedulers with RR Scheduler - Adaptive Modulation, Equal Distances

We also combine the comparisons of the performance of TCP-aware schedulers with that of RR and WB schedulers and plot the gain in $cwnd$ size and TCP throughput achieved in Figures 5.18 - 5.19. Similar to the results obtained with fixed modulation, we also observe that the improvement of TCP-aware scheduler over WB scheduler is more as compared to that of the improvement over RR scheduler. Since DTWUS-A scheduler performs better than that of TWUS-A scheduler, we only plot the comparisons of TWUS-A scheduler with that of RR and WB schedulers. Though we have illustrated the results for equal distance experiments, similar results are also valid for unequal distance experiments.

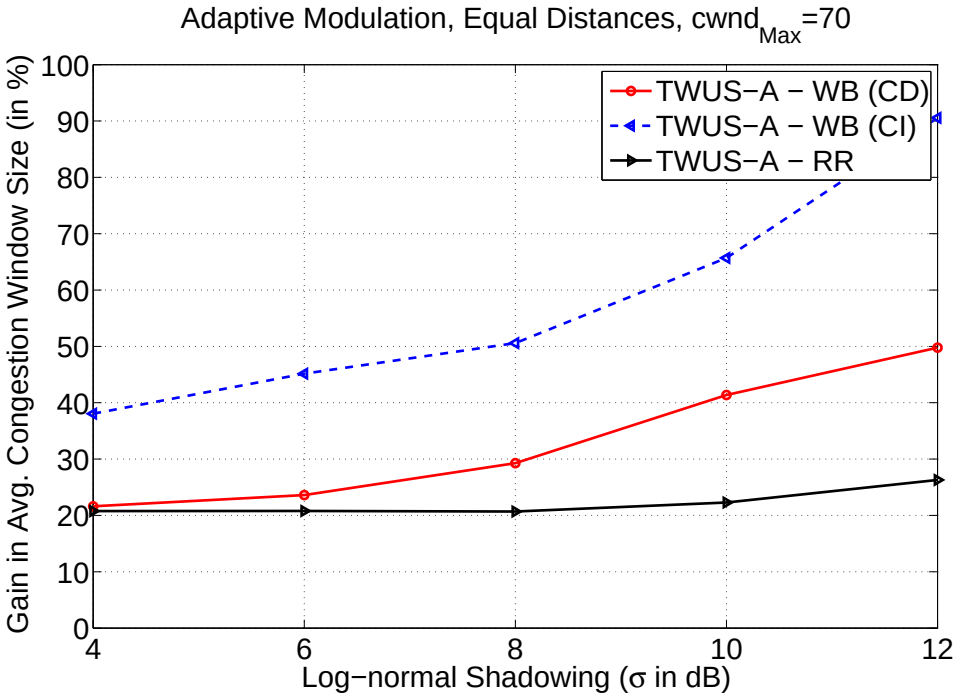


Figure 5.18: Gain in $cwnd$ size of TWUS-A over RR and WB Schedulers - Equal Distances

5.4.4 Adaptive Modulation vs. Fixed Modulation

From the simulation results presented in the previous section and in Section 4.4.1, we observe that the amount of data transmitted by the users in adaptive modulation scheme is more as compared to the amount of data transmitted in fixed modulation scheme. The average rate of transmission is almost double (80-90%) in adaptive modulation scheme as compared to fixed modulation scheme for similar conditions. The higher rate of transmission in adaptive modulation is achieved by adding extra complexity in the transmitter and receiver structures at SSs. Adaptive modulation scheme also increases average $cwnd$ size, resulting in higher TCP throughput. We

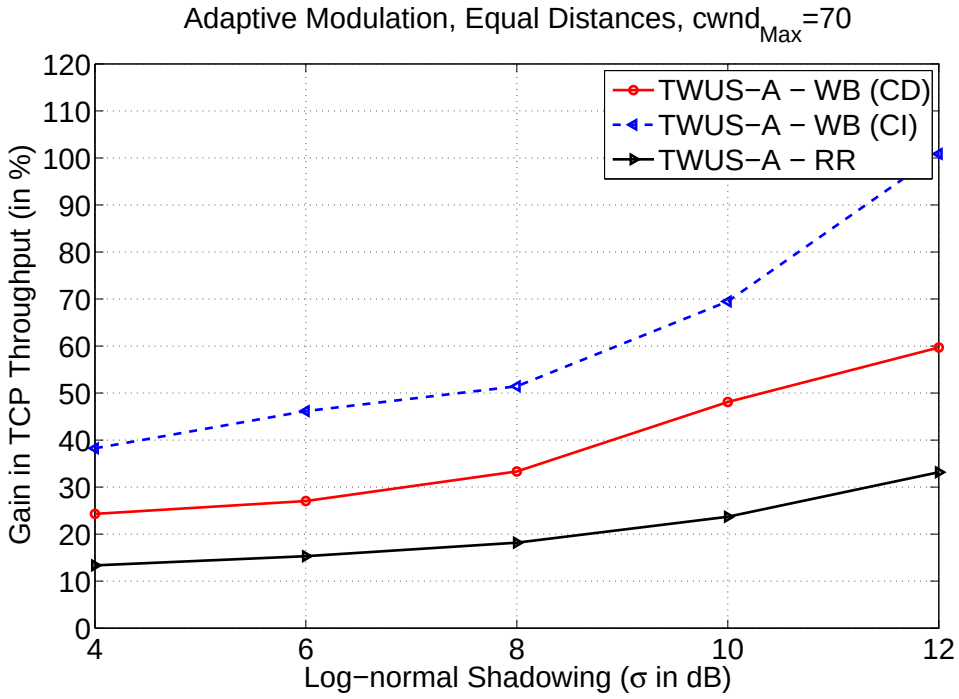


Figure 5.19: Gain in TCP Throughput of TWUS-A over RR and WB Schedulers - Equal Distances

also observe that the $cwnd$ size changes at a slower time scale than the transmission rate. Due to this reason, even though the rate of transmission increases by 80-90%, the average $cwnd$ size improves only by 50-55%. Similar observations are also valid for TCP throughput increase. On comparing the slot utilization of TCP-aware schedulers with fixed and adaptive modulations (Figures 4.15 - 4.16 and 5.12 - 5.13), we also observe that schedulers with fixed modulation scheme have higher slot utilization than schedulers with adaptive modulation scheme.

From both fixed as well as adaptive modulation experiments, we observe that as the standard deviation of the Log-normal shadowing increases, both average $cwnd$ and TCP throughput decreases. However, the rate of decrease of $cwnd$ and TCP throughput is more in adaptive modulation than that in fixed modulation. We also observe that schedulers with fixed modulation scheme have higher slot utilization than schedulers with adaptive modulation scheme.

5.4.5 Layer-2 and Layer-4 Fairness

In this section, we investigate both Layer-2 and Layer-4 fairness [66] of TCP-aware schedulers. We use Jain's Fairness Index as Layer-2 fairness index and use both Worst Case TCP Fairness Index (WCTFI) and TCP Fairness Index (TFI) as Layer-4 fairness index. The details of Layer-2

and Layer-4 fairness index have been discussed in Section 2.4.

Since the TCP-aware schedulers perform better than the RR and WB schedulers in terms of throughput and JFI, we present the Layer-2 and Layer-4 fairness only for the TCP-aware schedulers. Since TWUS-A and DTWUS-A outperform RR scheduler, we observe that both WCTFI and TFI are one for TWUS-A and DTWUS-A and hence the proposed scheduling schemes are fair at the Transport layer too.

5.5 TCP Throughput Analysis

In this section, we first develop an analytical framework to determine the throughput of TCP-based applications in a multipoint-to-point IEEE 802.16 network. The TCP throughput analysis is based on [75], a seminal work which models TCP Reno performance by the average TCP $send\ rate$ or throughput. In [75], the authors have derived a closed form expression for the steady state $send\ rate$ of TCP flow as a function of loss rate, RTT and TCP timeout. However, their model assumes a wired network where the packet losses are assumed to be due to congestion in the network. Modelling of TCP $send\ rate$ for wireless broadband networks like IEEE 802.16 requires the inclusion of the effect of MAC layer scheduling (which is absent in wired network) and packet losses due to the time varying nature of the channel. In this chapter, we incorporate the impact of scheduling at the MAC layer and wireless packet losses on the $send\ rate$ determination of TCP flows. We assume that the packets are of fixed sizes and are transmitted at the granularity of TCP Packets instead of MAC PDUs. This simplifies the complexity due to fragmentation and packing in our analysis. We also ignore the impact of ARQ mechanism for simplicity. Even though in the actual implementations of TCP, $cwnd_i$ increases by $\frac{1}{cwnd_i}$ each time an ACK is received, we do not consider this in our analysis. The reason is as follows: we assume that the $cwnd$ value does not change for one RTT. Based on the $cwnd_i$ value received during polling, the BS assigns time slots. Hence, even though the $cwnd_i$ change during a polling epoch, it will not be conveyed to the BS. The change in $cwnd$ size is only conveyed at the time of polling. We also assume that the time taken to transmit all packets of a window is smaller than one RTT. This means that there will be some idle periods, which we exploit during scheduling.

As discussed in Section 4.1.1, scheduling at the MAC layer has an impact on the $cwnd$

variation. Unless a SS is polled, it will not be considered for scheduling in that polling epoch. Unpolled SSs have to wait till the next polling epoch resulting delay in scheduling. We now determine the average uplink delay of the TCP-aware schedulers.

5.5.1 Average Uplink Delay of TCP-aware Scheduling

Let $cwnd_i(n)$ denote the congestion window of SS_i in frame n. We assume that the resource requirement of a SS does not change during a polling epoch (irrespective of whether a SS is polled or not). Let k denote the polling epoch in number of frames. As discussed before, a SS is successfully polled if it has a non zero $cwnd$ size and its SNR exceeds $\min\{SNR_{th}\}$. Let M denote the number of SSs polled successfully in a polling epoch. Let N denote the total number of SSs. We assume N to be fixed. The probability that M out of N users are polled at any polling instant is given by:

$$p(M) = \binom{N}{M} p^M (1-p)^{N-M}, \tag{5.8}$$

where p is the probability that $SNR_i \geq \min\{SNR_{th}\}$, for any SS_i. The number of SSs scheduled in a frame will always be less than or equal to M, the number of SSs in that polling epoch. The expected number of SSs polled successfully can be expressed as:

$$\begin{aligned} E[M] &= \sum_{M=0}^{N} Mp(M), \\ &= \sum_{M=0}^{N} M\binom{N}{M} p^M (1-p)^{N-M}. \end{aligned} \tag{5.9}$$

After determining the expected number of SSs to be polled in any polling epoch, we determine the number of frames that a SS needs to wait before getting polled. In TCP-aware scheduling, if one SS misses polling, it needs to wait again for k frames before obtaining another chance of polling. Though the value of polling epoch k depends on the RTT of the flows, we assume fixed k in this section for simplicity. SS_i gets polled successfully in the first polling epoch with probability p, otherwise, it gets polled in the second polling epoch with probability $(1-p)p$ and so on. Hence, the expected number of polling epochs L is:

$$E[L] = \sum_{L=1}^{\infty} Lp(1-p)^{L-1} - 1. \tag{5.10}$$

The expected number of frames that a SS waits before getting polled successfully is $E[L] \times k \times T_f$. Once a SS is polled successfully, it gets scheduled in the first frame of the polling epoch, but it may or may not be scheduled in the subsequent frames, since in every

frame the BS requires the SNR to exceed $\min\{SNR_{th}\}$ before scheduling. In the successive frames, M' out of M polled users are scheduled and the value of M' varies from frame to frame ($M' \leq M \leq N$). The expected number of SSs successfully scheduled out of M polled users is determined as:

$$E[M'/M] = \sum_{M'=0}^{M} M' \binom{M}{M'} p^{M'}(1-p)^{M-M'}, \tag{5.11}$$

and the expected number of users scheduled in a frame is expressed as:

$$\begin{aligned} E[M'] &= \sum_{M=0}^{N} E[M'/M]p(M), \\ &= \sum_{M=0}^{N} \left(\sum_{M'=0}^{M} M' \binom{M}{M'} p^{M'}(1-p)^{M-M'} \right) p(M). \end{aligned} \tag{5.12}$$

5.5.2 Determination of Uplink Scheduling Delay

After determining the expected number of SSs successfully scheduled in each frame, we determine the average number of frames in which a polled SS gets scheduled. On an average, due to TCP-aware scheduling a polled SS transmits in $\frac{E[M']}{E[M]} \times k$ frames out of k frames in a polling epoch. Hence, the average time a polled SS remains idle (not scheduled) due to TCP-aware scheduling is expressed as:

$$T_{wf} = \left(1 - \frac{E[M']}{E[M]}\right) kT_f. \tag{5.13}$$

Thus, the extra time that a SS waits due to scheduling in the uplink can be expressed as the sum of polling delay and idleness during a polling epoch. The average uplink delay due to TCP-aware scheduling can be expressed as:

$$T_{uld} = E[L] \times k \times T_f + T_{wf}. \tag{5.14}$$

5.5.3 TCP $send\ rate$ Determination

If a SS is polled successfully, then it has a chance of being scheduled in that polling epoch. It is possible that the SS may not be able to transmit one window worth of data or may experience triple *dupacks* or timeouts in that polling epoch. However, if a SS is not polled, the chance of

increasing its congestion window is minimal. Increase in congestion window only occurs, if it receives a pending ACK.

From the above discussion, it is evident that TCP behavior does not change due to TCP-aware scheduling. Instead it follows the behavior seen in wired network, except for the extra delay due to polling (not due to scheduling) and wireless packet losses. The wireless packet losses can be neglected due to adaptive modulation and rate adaptation in our framework. Thus the only extra factor that we need to consider for TCP $send\ rate$ determination is the polling delay.

As discussed in [75], the average TCP $send\ rate$ in a wired network can be expressed as:

$$B_w \approx \min\left(\frac{cwnd_{Max}}{RTT_w}, \frac{1}{RTT_w\sqrt{\frac{2bp_w}{3}} + TO\min\left(1, 3\sqrt{\frac{3bp_w}{8}}\right)p_w(1+32p_w^2)}\right), \tag{5.15}$$

where B_w is the TCP end-to-end throughput for a wired network (in packets per unit time), b is the number of TCP packets acknowledged by one ACK, RTT_w is the average TCP round trip time in a wired network (only due to congestion), p_w is the wirelink loss probability and TO is the average TCP timeout value. Since TCP does not differentiate between wireless loss and congestion loss, we assume all losses are congestion losses and determine the number of loss indications. In TCP, since the congestion window size can grow upto $cwnd_{Max}$, maximum TCP $send\ rate$ is bounded by $\frac{cwnd_{Max}}{RTT_w}$.

The end-to-end average TCP throughput or $send\ rate$ for IEEE 802.16 based network can be expressed by incorporating polling delay. The round trip time RTT_{wr} in Eqn. (5.15) can be modified as:

$$RTT_{wr} = RTT_w + E[L] \times k \times T_f. \tag{5.16}$$

By replacing RTT_w by RTT_{wr} in Eqn. (5.15), end-to-end TCP throughput or $send\ rate$ for IEEE 802.16 based network (B_{wr}) can be expressed as:

$$B_{wr} \approx \min\left(\frac{cwnd_{Max}}{RTT_{wr}}, \frac{1}{RTT_{wr}\sqrt{\frac{2bp_w}{3}} + TO\min\left(1, 3\sqrt{\frac{3bp_w}{8}}\right)p_w(1+32p_w^2)}\right). \tag{5.17}$$

5.5.4 Validation of TCP Throughput

To validate the TCP throughput analysis and average $send\ rate$ of TCP flows, we compare the average TCP $send\ rate$ obtained in Eqn. (5.17) with our simulation results. We determine the probability of loss (p_w) similar to that of [75]. We consider both triple-duplicate ACKs and TCP timeouts as loss indications. Let p_w be the ratio of the total number of loss indications to the total number of packets transmitted. From simulations, we observe that the average probability (p) that $SNR_i \geq \min\{SNR_{th}\}$ is 0.87, $\forall i \in I$. Using Eqn. (5.16), we determine RTT_{wr}. Then by using the value of RTT_{wr} obtained using Eqn. (5.16) and p_w obtained above, we determine the average TCP $send\ rate$ (analytical) using Eqn. (5.17). The average uplink scheduling latency for various channel conditions and equal (E) and unequal (U) distances are presented in Figures 5.20 - 5.23.

To verify our model, we determine average TCP $send\ rate$ for all four sets of experiments (TWUS-A with equal and unequal distances, DTWUS-A with equal and unequal distances). We plot the analytical and experimental TCP throughput at different $cwnd_{Max}$ in Figures 5.24 - 5.27. From these figures, we observe that the theoretical $send\ rate$s determined using Eqn. (5.17) and the $send\ rate$ obtained by our simulations are matching very closely.

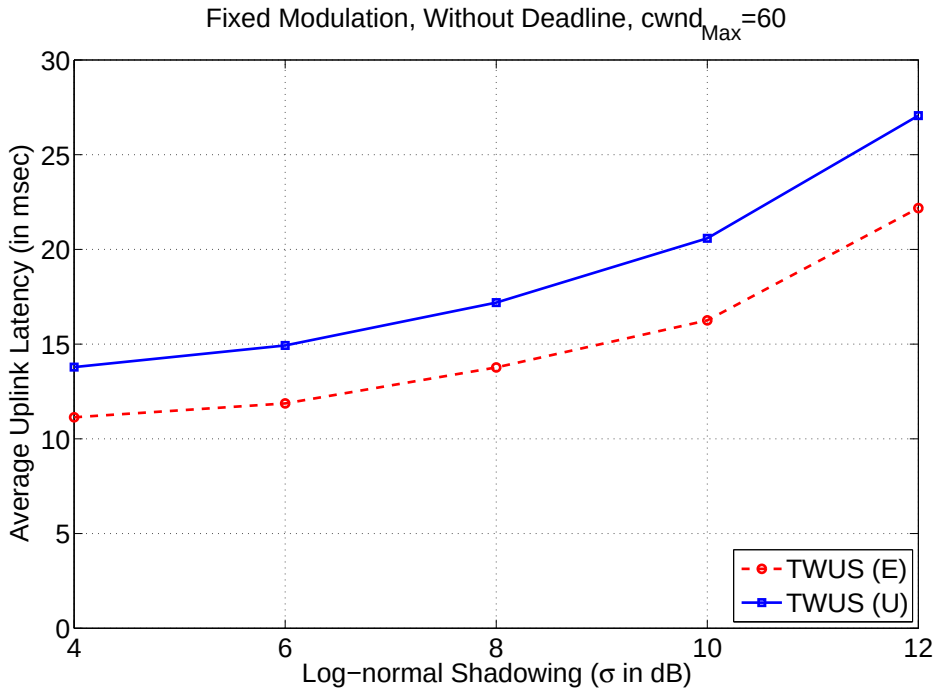


Figure 5.20: Average Uplink Scheduling Latency of TWUS

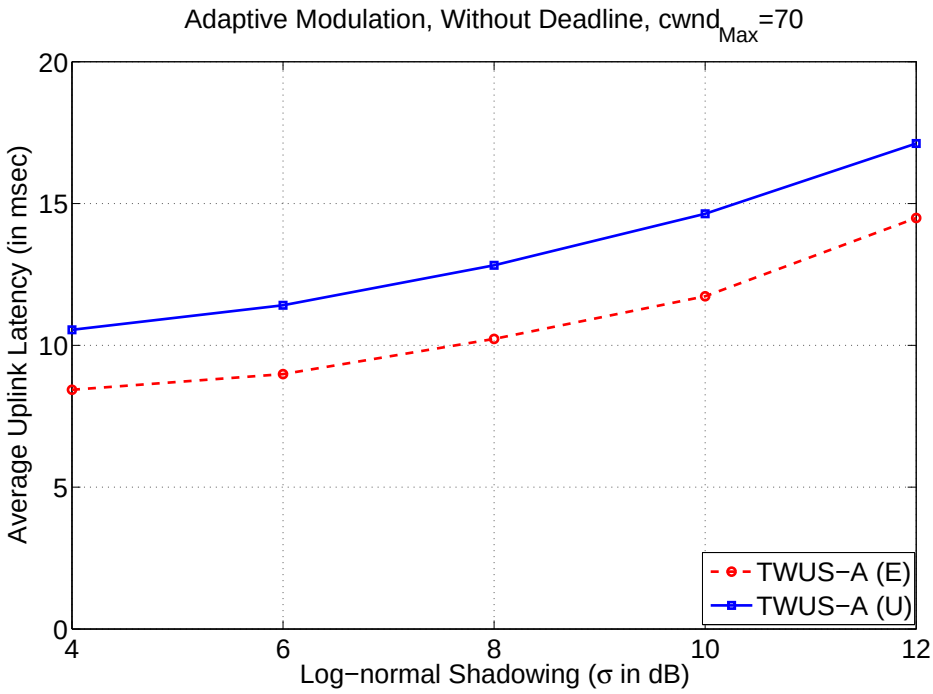


Figure 5.21: Average Uplink Scheduling Latency of TWUS-A

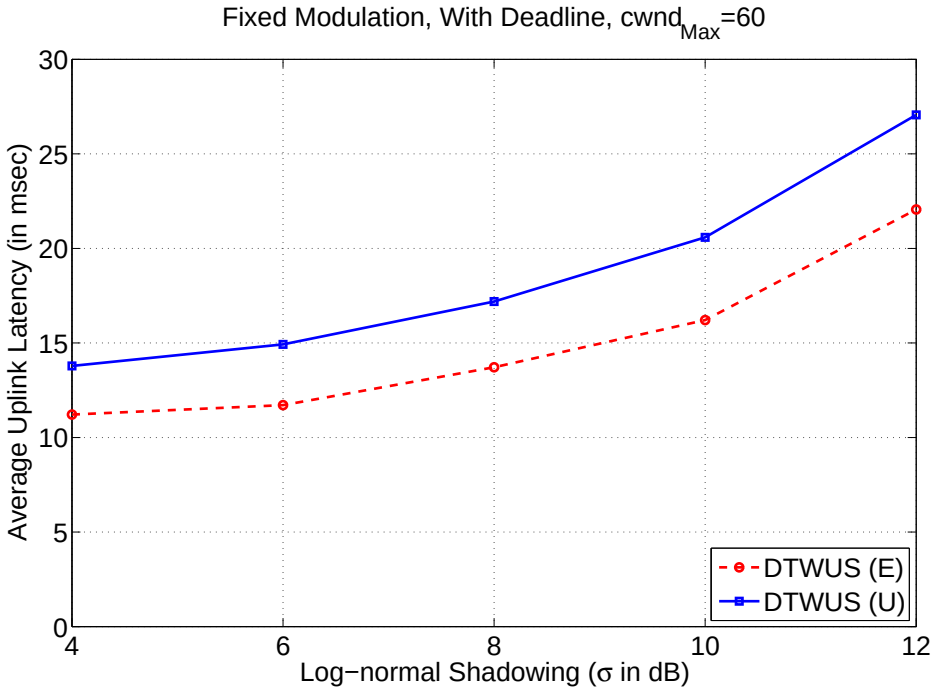


Figure 5.22: Average Uplink Scheduling Latency of DTWUS

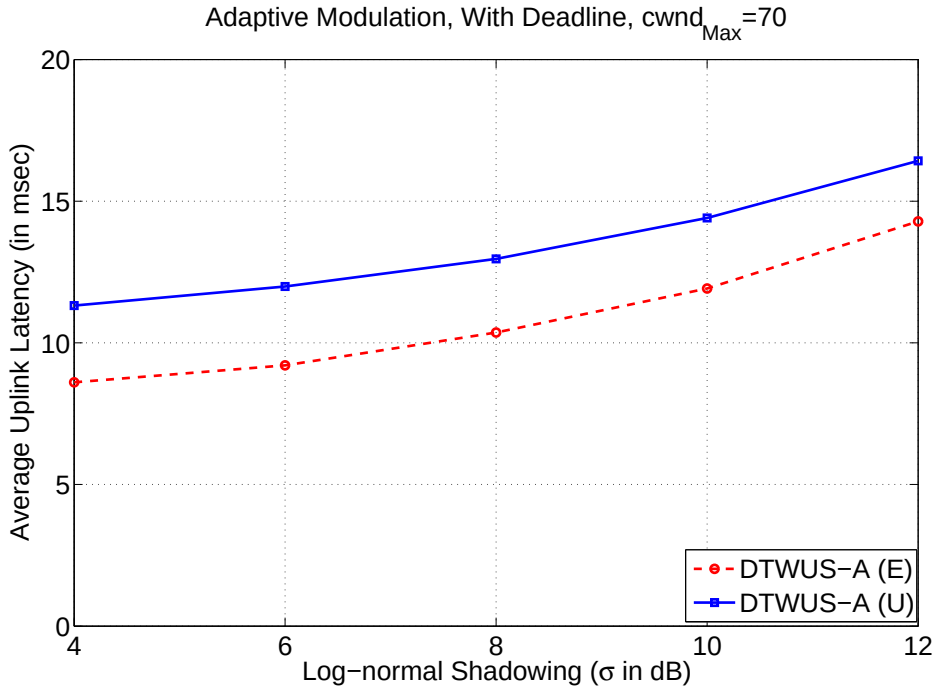


Figure 5.23: Average Uplink Scheduling Latency of DTWUS-A

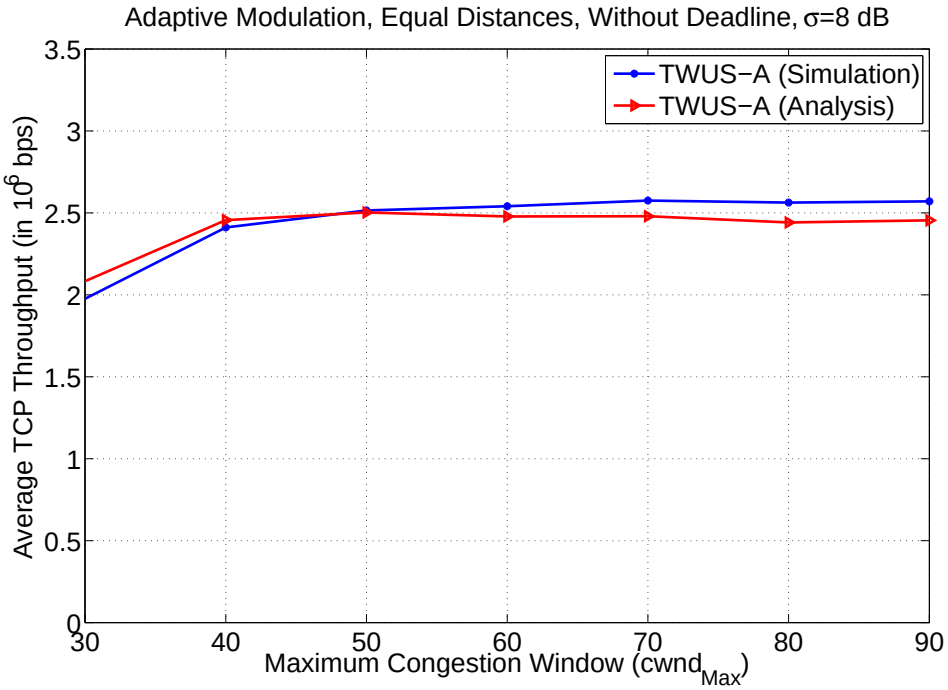


Figure 5.24: Average TCP Throughput of TWUS-A at Different $cwnd_{Max}$ - Equal Distances

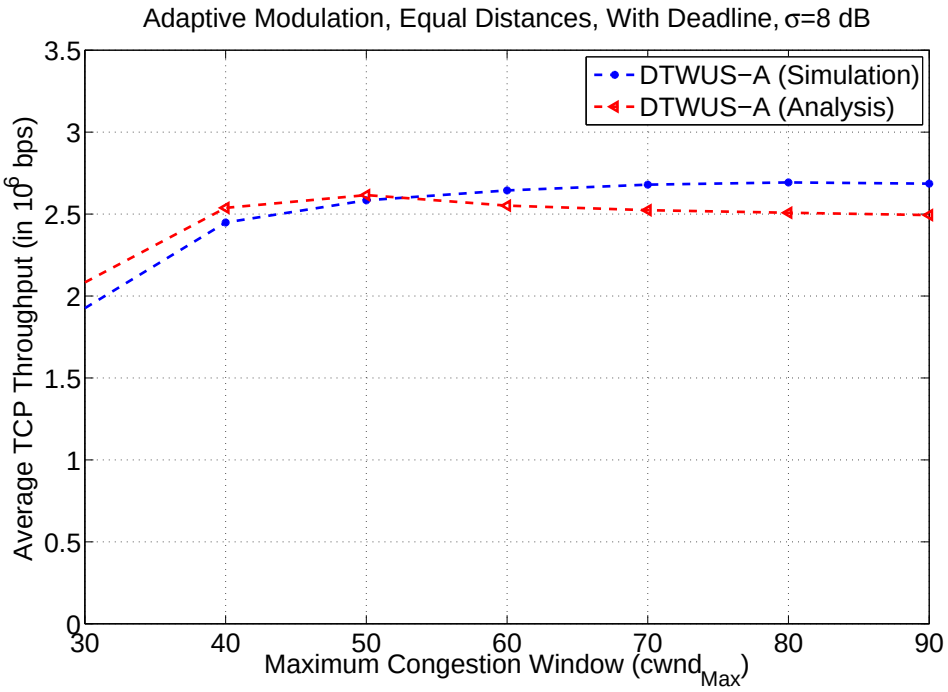


Figure 5.25: Average TCP Throughput of DTWUS-A at Different $cwnd_{Max}$ - Equal Distances

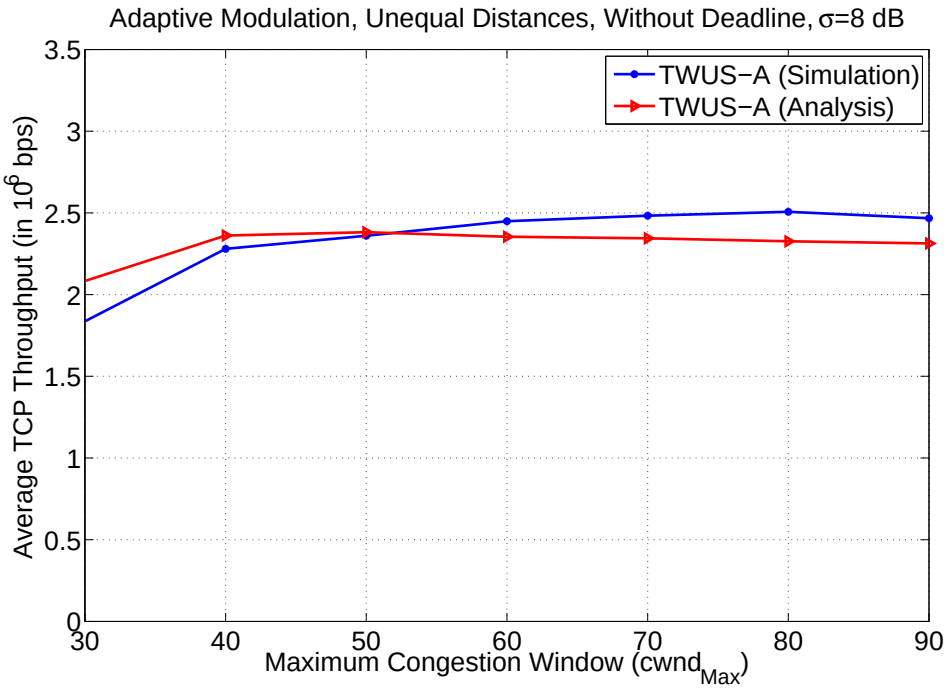


Figure 5.26: Average TCP Throughput of TWUS-A at Different $cwnd_{Max}$ - Unequal Distances

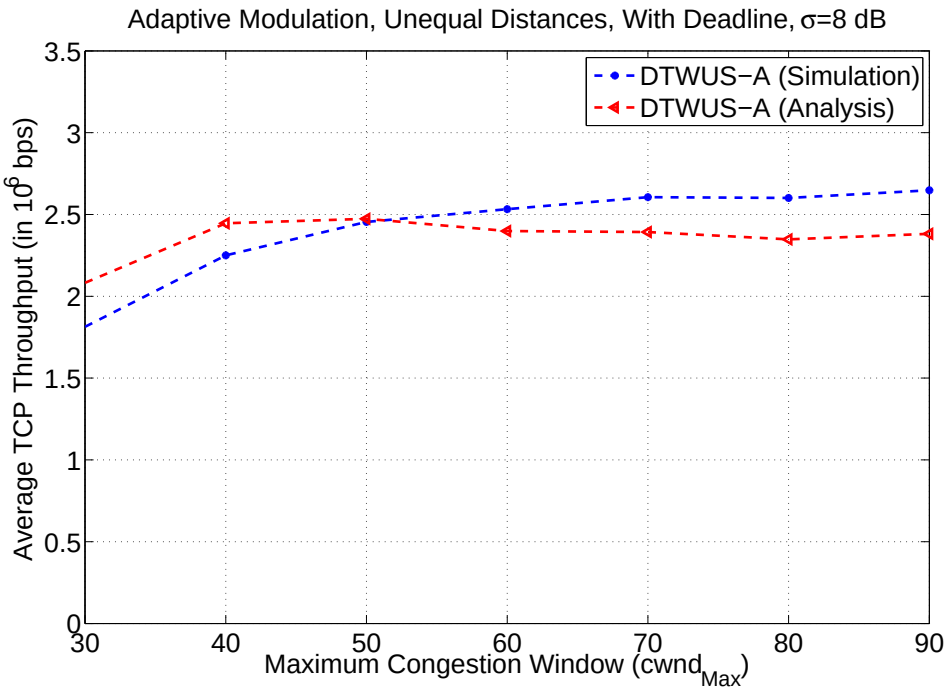


Figure 5.27: Average TCP Throughput of DTWUS-A at Different $cwnd_{Max}$ - Unequal Distances

It has been investigated that the DRR scheduler proposed for a wired network is a long-term fair scheduler [29]. Since the TCP-aware scheduler is a variant of DRR scheduler, it should follow the basic properties of DRR scheduler. However, the TCP aware scheduler operates in a multipoint-to-point wireless network and uses a polling method to select the list of schedulable users. Therefore, even though it is a variant of DRR scheduler, its implementation requirements are different from that of DRR scheduler. We therefore, investigate its properties in the next section.

5.6 Properties of TCP-aware Schedulers

Since the TCP-aware scheduler with fixed modulation can be considered as a special case of TCP-aware scheduler with adaptive modulation, we investigate the properties of TCP-aware scheduler with adaptive modulation. Before analyzing the properties of TCP-aware scheduler with adaptive modulation, we first investigate the properties of quantum size $Q(n)$ and deficit counter $DC_i(n)$.

Property 1. *The maximum value of quantum size $Q(n)$ is bounded by $cwnd_{Max}$.*

In a polling epoch, since the resource requirement of a SS cannot exceed the maximum

congestion window size ($cwnd_{Max}$) defined by the protocol, the average amount of data transmission per SS in a frame cannot exceed $cwnd_{Max}$. Hence, the quantum size $Q(n)$ is bounded by $cwnd_{Max}$. This can be verified from Eqn. (5.2) also.

Property 2. *Deficit Counter DC_i, $\forall i \in I$ must satisfy: $|DC_i| \leq cwnd_{Max}$*

At the beginning of the polling epoch, the deficit counter $DC_i(0) = 1$. During a polling epoch, the value of the deficit counter can become either positive or negative. From Eqn. (5.3), we observe that if a schedulable SS is not assigned any slots in a polling epoch, then its deficit counter will rise to the maximum value of the quantum size $Q(n)$. Since $Q(n)$ is limited by $cwnd_{max}$, the deficit counter is also limited by $cwnd_{Max}$.

Contrary to the above, if SS_i is scheduled in one polling epoch such that its entire window of data is transmitted, then the resource requirement of SS_i drops to zero. This will result in the removal of SS_i from the active list. In this case, the deficit counter can reach $-cwnd_{Max}$, just before SS_i is withdrawn from the active set.

As discussed in Chapter 2, a good scheduling algorithm must guarantee fair share of the network resources to each user over a certain time duration. For example, TCP-based applications are unfair over one RTT. This is because, the rate of transmission of TCP-based application is governed by its $cwnd$ size. Since $cwnd$ sizes are different for different users over one RTT, the amount of packet transmitted by the TCP-based applications are different for different users. This results in unfairness.

Therefore, the minimum duration over which TCP-aware schedule is fair depends upon the polling epoch and TCP timeout value. Since in a polling epoch, the proposed scheduler schedules the polled SSs only, any SS that misses polling will have to wait for the next polling. If p is the probability that a SS satisfies the SNR condition for polling (assuming all SSs have non-zero $cwnd$ size), then we can show that for $p \geq 0.5$ (a likely situation), the expected number of polling epochs $E[L]$ which one SS waits before successfully being polled is less than 2 (can be verified from Eqn. (5.10)). Hence, we can argue that the TCP-aware scheduler polls any backlogged SS on an average of two RTTs. Therefore, it is quite likely that all backlogged SSs get polled before the timeout (which corresponds to approximately four RTTs) occurs. Hence, we consider the average TCP timeout (TO) as the minimum duration over which fairness needs to be investigated.

We now investigate the fairness properties of TCP-aware schedulers.

5.6.1 Fairness Measure of TCP-aware Scheduler

Let SS_i and SS_j be two backlogged subscriber stations in time interval (t_1, t_2). Let $Tx_i(t_1, t_2)$ and $Tx_j(t_1, t_2)$ denote the amount of data in bytes transmitted by SS_i and SS_j respectively in this interval. Let $FM(t_1, t_2)$ denote the maximum difference of data transmitted by any two backlogged pair of SSs. This can be expressed as:

$$FM = \max_{\forall i,j} |(Tx_i(t_1, t_2) - Tx_j(t_1, t_2))|, \forall (t_1, t_2). \tag{5.18}$$

The scheduling process is fair, if the fairness measure FM is bounded by a small constant irrespective of the intervals. The fairness measure and the time interval defined here is in general applicable to wired networks. In a wireless network, since the channel is time varying, we define expected fairness guarantee instead of absolute fairness guarantee. As discussed before, the minimum duration over which we define fairness measure is the average TCP timeout (TO) of the flows. We define the wireless fairness measure as follows:

$$FM_w = \max_{\forall i,j} E|(Tx_i(t_1, t_2) - Tx_j(t_1, t_2))|, \forall (t_2 - t_1) \geq TO. \tag{5.19}$$

Theorem 1. *For an interval $(t_1, t_2) \geq TO$, the wireless fairness measure FM_w is bounded by a small constant.*

Proof. Let the interval (t_1, t_2) consists of m frames. Let m_i denote the number of frames in which SS_i transmits and m_j denote the number of frames in which SS_j transmits within the interval (t_1, t_2). Let $Tx_i(n)$ denote the amount of data transmitted by SS_i in frame n. We define $Tx_i(t_1, t_2)$ as the cumulative sum of the amount of data transmission on all m_i frames during the interval (t_1, t_2). In other words,

$$Tx_i(t_1, t_2) = \sum_{n=1}^{m_i} Tx_i(n). \tag{5.20}$$

From the deficit counter update defined in Eqn. (5.3), we write

$$\begin{aligned} DC_i(n) &= DC_i(n-1) + Q(n) - Flag_i \times R_i(n-1) \times N_i(n-1) \\ &= DC_i(n-1) + Q(n) - Tx_i(n-1). \end{aligned} \tag{5.21}$$

Using Eqn. (5.21) we re-write Eqn. (5.20) as follows:

$$\begin{aligned} Tx_i(t_1,t_2) &= \sum_{n=1}^{m_i} Tx_i(n), \\ &= \sum_{n=0}^{m_i-1} DC_i(n-1) - DC_i(n) + Q(n), \\ &= \sum_{n=0}^{m_i-1} Q(n) + DC_i(0) - DC_i(m_i-1), \\ &\leq m_i \times cwnd_{Max} + DC_i(0) - DC_i(m_i-1), \\ &\leq m_i \times cwnd_{Max} + 1 - DC_i(m_i-1). \end{aligned} \tag{5.22}$$

Since the channel gains of all subscriber stations are i.i.d., and both SS_i and SS_j are backlogged, the polling and scheduling opportunities provided by the TCP-aware scheduler to both SS_i and SS_j are similar. Hence, the expected number of frames employed by SS_i and SS_j transmission are similar. Hence, we assume that $E[m_i] = E[m_j]$. Since $|DC_i(m_i-1)| \leq cwnd_{Max}$, we can express the fairness measure (FM_w) as follows:

$$\begin{aligned} FM_w &= \max_{\forall i,j} E[|(Tx_i(t_1,t_2) - Tx_j(t_1,t_2))|], \\ &\leq E[(m_i \times cwnd_{Max} + 1 - DC_i(m_i-1)) - (m_j \times cwnd_{Max} + 1 - DC_j(m_j-1))], \\ &\leq 2 \times cwnd_{Max}, \end{aligned} \tag{5.23}$$

which is a constant and a finite number, irrespective of the length of the interval. For a large time interval, the value of this constant is very small as compared to the amount of data transmitted by any backlogged user. □

Part II

Channel aware Congestion Control in Infrastructure-less Ad-hoc Networks

Chapter 6

Congestion Control in Wireless Ad-hoc Networks

We have discussed TCP-aware fair uplink scheduling schemes for a multipoint-to-point network in Chapters 4 and 5. These schemes operate at the MAC layer of the protocol stack and facilitate the TCP-based applications to achieve higher throughput without altering the congestion control mechanism of TCP. These are centralized uplink scheduling schemes and are cross-layer in nature, involving PHY, MAC and Transport layers. However, these scheduling schemes are not applicable in de-centralized wireless networks, such as wireless ad-hoc networks, as there is no central entity involved for scheduling.

In ad-hoc networks, in addition to the location dependent time varying nature of wireless channel, interference and limited energy of wireless nodes also pose challenges in designing schemes to provide higher throughput. For the best effort services, Transmission Control Protocol (TCP) based congestion control technique plays a significant role while designing these systems. This is because, TCP treats packet drops as an indication of congestion. However, in wireless networks packet drops can occur not only because of congestion in the network, but also due to interference and wireless channel characteristics. This can be alleviated either by: (1) modifying TCP congestion control mechanism, taking the nature of wireless channel and network model into consideration or by (2) exploring channel aware congestion control mechanism. Since TCP is a popular protocol, modification of TCP to alleviate packet drops due to interference and channel characteristics is not advisable. Instead, this can be alleviated by channel aware congestion control techniques.

This chapter is organized as follows. In section 6.1, we discuss various congestion control techniques used in wired networks. In Section 6.2, we model congestion control of wired network using an optimization framework. We review the extension of congestion control techniques proposed for wired network to wireless networks in Section 6.3. In Section 6.4, we discuss the need of cross-layer congestion control technique, model an optimization-based congestion control technique and discuss the nature of the utility function of TCP NewReno. We discuss some of the open problems in cross-layer based congestion control in ad-hoc networks in Section 6.5.

6.1 Congestion Control in Wired Networks

In Section 4.1, we have discussed TCP and its window evolution. In this section, we discuss some congestion control techniques used in wired networks. This section presents a brief review of published techniques and terminology, which we use in the rest of the chapter. Comprehensive account of congestion control techniques can be found in [42, 43, 72, 76–78].

In a wired network, the links are assumed to be reliable and of fixed capacities. Therefore, any packet loss, delay or out of order delivery of packets is mainly due to congestion in the network. Congestion in the network can be detected by the TCP of the protocol stack. To counter congestion in the network, TCP retransmits lost packets, rearranges out-of-order delivered packets and adapts its rate of transmission by controlling its congestion window ($cwnd$) size. Congestion control techniques of TCP can be broadly divided into *Congestion Control* and *Congestion Avoidance*. In congestion control [79], TCP controls the $cwnd$ size as the network parameter *after* sensing congestion in the network (*reactive*); whereas, in congestion avoidance, TCP controls the $cwnd$ size as the network parameter *before* it experiences congestion (*proactive*). The congestion control techniques are further divided into three broad categories, (i) *window-based*, (ii) *equation-based*, and (iii) *rate-based*, which are discussed in the following sections.

6.1.1 Window-based Congestion Control

The congestion control algorithm followed by most of the variants of TCP is an end-to-end window-based congestion control technique [72], which employs $cwnd$ size as the network

parameter. These variants of TCP follow an *Adaptive Window Management* technique in which the sources increase their $cwnd$ sizes on reception of successful ACK packets. Contrary to that when the TCP sources receive multiple ACKs of a same packet in succession or do not receive any ACK before the expiry of a timeout, they decrease their $cwnd$ sizes. In TCP, instead of using linear increase and decrease of $cwnd$, Additive Increase and Multiplicative Decrease (AIMD) method is being used. Moreover, the manner of increase and decrease of $cwnd$ size is different for different TCP implementations. We discuss some of the commonly used implementations as follows:

TCP Tahoe: In TCP Tahoe [80], congestion is detected by timeouts only. On timeouts, the TCP source decreases $cwnd$ size to one. This phase of TCP Tahoe is known as *fast retransmit*. After it drops its $cwnd$ size to one, it increases its $cwnd$ size from one to ss_{thresh} (which acts as limit point) in an exponential manner in each RTT. This phase is known as *slow start* phase. After reaching ss_{thresh}, the source increases its $cwnd$ size linearly in each RTT till it experiences congestion. This phase is known as *congestion avoidance* phase.

TCP Reno: In TCP Reno [73], congestion is detected by both triple *dupacks* and timeouts. On timeouts it works similar to TCP Tahoe. However, with triple *dupacks*, it follows *fast recovery* technique and decreases its $cwnd$ value by half of its current size and then increases linearly until it experiences congestion. Though it is better than TCP Tahoe for dealing with single packet loss and triple *dupacks*, the performance suffers when multiple packets are lost within one RTT. This problem is addressed in its later version called NewReno or TCP Reno-2.

TCP NewReno: TCP NewReno (TCP Reno-2) [74, 81] is an extension of TCP Reno. It follows exactly as TCP Reno except for the multiple packet drops in one RTT. In NewReno, $cwnd$ value is not decremented for every packet loss, instead it is decremented in an intelligent manner. In this scheme, for multiple packet drops within one RTT (multiple packet losses from a single window of data), instead of dropping its $cwnd$ size by multiple times, the TCP source drops only once in that RTT.

TCP Vegas: In TCP Vegas [82, 83], packet delay rather than packet loss is used as an indication of congestion in the network. In this scheme, the TCP sources increase or decrease their $cwnd$ sizes based on the difference between the expected throughput (ratio of current $cwnd$ size and the minimum RTT measured, which is known as *baseRTT*) and actual throughput (ratio of number of bytes transmitted during the last RTT, known as *rttLen* and averaged RTT

of the last segment). In this scheme, TCP sources decrease their $cwnd$ sizes before experiencing any congestion in the network in a more aggressive manner than Reno and Tahoe. TCP Vegas is proactive, whereas TCP Tahoe, Reno and NewReno are reactive.

6.1.2 Equation-based Congestion Control

In window-based congestion control, the TCP sources adapt their $cwnd$ sizes using AIMD technique. However, adaptation of $cwnd$ sizes will have an impact on the performances when the applications are played real-time. This is because, if the TCP sources drop their $cwnd$ sizes in a multiplicative manner (either by half of its current value or to one) on every congestion indication they experience, then there will be abrupt changes on the send rate (transmission rate) of the applications, which will have a noticeable impact on the performance of the application. Instead, to maintain the quality of reception, one needs to maintain a relatively steady send rate, while still being responsive to congestion. The equation-based congestion control [84] adapts its send rate based on the loss event rate, which in turn depends upon the number of packets dropped within a single RTT. It uses a control equation that explicitly gives the maximum acceptable send rate as a function of the recent loss event rate. Instead of reducing the rate of transmission drastically using any AIMD technique, it reduces its rate of transmission guided by a control equation. It increases its send rate in a slow rate, in response to a decrease in loss event rate.

6.1.3 Rate-based Congestion Control

The analysis in [85], shows that the instantaneous throughput of a connection is inversely proportional to the square of round-trip delay of the connection. The window-based congestion control is unfair towards the connections with higher round-trip delays. When multiple connections share a bottleneck link and the delay-bandwidth product is large, adapting the rate of transmission using AIMD technique can lead to underutilization of bandwidth. Since the delay of the network is very high, the sources will take long time to reach the available bandwidth. Moreover, unfairness will result if flows have different RTTs.

The unfairness and the underutilization of the network bandwidth can be alleviated, if the delay-bandwidth product is lesser than the buffer in the link. In that case, the instantaneous throughput or rate of transmission of an user can be approximated by $x_i = \frac{cwnd_i}{RTT_i}$, where x_i is

the instantaneous rate, $cwnd_i$ is the window size and RTT_i is the RTT of user i at that instant. Hence, for a high speed network, the congestion control can be modeled as rate-based instead of usual window-based one. Instead of changing the window size in each RTT, instantaneous rates can be regulated at a smaller time scale to avoid congestion. By regulating the instantaneous rates in this manner, maximum bandwidth can be utilized.

Rate-based congestion control scheme has been further investigated in [86]. In [86], the authors formulate end-to-end congestion control as a global optimization problem of a class of Minimum Cost Flow Control (MCFC) algorithms for controlling the rate of a connection/session or window size. The window-based congestion control of TCP is shown as a special case of MCFC algorithm in this paper.

Along with the above categories of congestion control schemes, Adaptive Queue Management (AQM) based schemes are also employed to control congestion in the network. Schemes like Random Early Detection (RED) [87] and its variants (Stabilized RED (SRED) [88], Adaptive RED (ARED) etc.) and Explicit Congestion Notification (ECN) [34] are also used to control congestion in a proactive manner. In these schemes, along with the end-to-end principle of TCP, network feedback is used for congestion detection. In the next section, we discuss ECN and its use in brief.

6.1.4 Explicit Congestion Notification

It is a congestion avoidance technique where the sink indicates congestion before the on set of actual congestion in the network. This is achieved by the AQM mechanism which allows routers to set the Congestion Experienced (CE) code-point in a packet header as an indication of congestion (instead of relying solely on packet drops). This has the potential of reducing the impact of loss on delay-sensitive flows. This technique is known as Explicit Congestion Notification (ECN) [34]. ECN for wired network is a well established technique. There are few studies [89, 90], which exploit the ECN marking of wired network to control the random losses due to the time varying nature of wireless channel. But, these models do not consider losses due to fading and they do not distinguish the packet losses due to time varying nature of the channel and due to congestion in the network. Further studies are required in this direction such that ECN can distinguish between wireless channel loss and congestion loss, and to model a flow control problem of window size in wireless ad-hoc networks. This can improve the end-to-end

throughput of wireless networks.

6.2 Congestion Control using Optimization Framework

Modelling congestion control as an optimization flow control problem has been addressed in [33, 79, 91–93]. These articles have modelled congestion control as a constraint based optimization problem. Though all of them model congestion control problem as a social optimization problem, each has a different solution approach. In this section, we discuss two specific approaches such as, *Charging and Rate Control* [91] or *Kelly's Model* and *Optimization Flow Control* [33] or *Low and Lapsley Model*, to solve the constraint based optimization problem and the congestion in the network. Before discussing the solution methodologies, we explain the system model in the following section.

6.2.1 System Model and Problem Formulation

We consider a network of N communicating nodes connected by L links. The links are indexed by $l \in L$ and have finite capacity c_l packets/sec. Let a set of I source-sink pairs (flows) share the network of L unidirectional links. Let each source-sink pair $i \in I$ be associated with a transmission rate x_i. Let $U_i(x_i)$ be the utility associated with the transmission rate x_i. We assume that the utility function is concave, increasing and double differentiable [94] for elastic[1] traffic. Let H denote the routing matrix comprising of the route information of all possible links. We assume stationary routing in our analysis. The elements of the routing matrix H are expressed as:

$$H_{li} = \begin{cases} 1 & \text{if, source-sink pair } i \text{ uses link } l., \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

We define the aggregate flow y_l (packets/sec) at link l as:

$$y_l = \sum_i H_{li} x_i. \tag{6.2}$$

[1]Elastic traffic consists of traffic for which users do not necessarily have a minimum requirements, but would get as much as data through to their respective destinations as quickly as possible [95].

Let μ_l be the cost of transmission associated with link l. The cost of transmission μ_l can be considered as the congestion cost of the link. The total cost associated with the source-sink pair i is expressed as:

$$q_i = \sum_l H_{li}\mu_l. \tag{6.3}$$

Both [91] and [33] consider the objective of maximizing the aggregate utility of all sources (or system utility) subject to linear capacity constraints of the links. This corresponds to a system/social optimality and is expressed as:

$$\begin{aligned} &\max_{x_i \geq 0} \sum_i U_i(x_i), \\ \text{s. t.,} \quad &\sum_{i \in I(l)} H_{li}x_i \leq c_l; \\ &X = \{x_i\},\ X \geq 0. \end{aligned} \tag{6.4}$$

Solving Eqn. (6.4) centrally would require the knowledge of all utility functions and coordination among all sources as the sources share the links. Since this is not possible in practical networks, various de-centralized solutions have been considered. We discuss such different de-centralized approaches in the subsequent sections.

6.2.2 Charging and Rate Control

In this model, a user (source-sink pair) chooses the charge per unit time that it can pay. The network determines the rate at which the user should transmit according to a proportional fairness criterion applied to the rate per unit charge. A system equilibrium is achieved when the users choice of charges and networks choice of allocated rates are in equilibrium. The system comprises both the users with utility functions and a network of links with capacity constraints. As discussed in [91], the system optimization problem $SYSTEM(U, H, C)$ is identical to that of Eqn. (6.4). Solution to the system optimization problem is obtained by solving two subproblems; one at the user level, known as $USER(U_i(x_i), q_i)$ and the other at the network level, known as $NETWORK(H, C; q)$. $USER(U_i(x_i), q_i)$ is also known as user optimization problem, whereas $NETWORK(H, C; q)$ is known as revenue optimization problem. These two subproblems are expressed as:

$USER(U_i(x_i), q_i)$

$$\max_{x_i \geq 0} U_i(x_i) - q_i x_i, \tag{6.5}$$

where q_i is the charge per unit flow that user i is willing to pay. The user is allowed to vary x_i freely.

$NETWORK(H, C; q)$

$$\begin{aligned} &\max_{X \geq 0} \sum_i q_i x_i, \\ \text{s. t.,} \quad &HX \leq C; \\ X = \{x_i\} \quad &\text{and} \quad C = \{c_l\} \end{aligned} \tag{6.6}$$

Since there exists a unique q_i that solves $USER(U_i(x_i), q_i)$ and at the same time, there exists an optimal $X = \{x_i\}$ for the charge that the network received $q = \{q_i\}$ by solving $NETWORK(H, C; q)$, $SYSTEM(U, H, C)$ has a unique solution.

6.2.3 Optimization Flow Control

Solving Eqn. (6.4) not only requires the knowledge of utility function of all users, but also requires coordination of all users. Since in a de-centralized network, obtaining the knowledge of the utility function of all users and coordination among all users is not possible, [33] proposes de-centralized solution to determine the source rate x_i of each users. This solution is a reactive flow control technique, where each link $l \in L$ determines a price μ_l for a unit bandwidth in l, based on all local source rate x_i at that link. A source $i \in I$ is fed back with the total price in its path $q_i = \sum_{l \in L} H_{li}\mu_l$, which we illustrate in Figure 6.1. Therefore, the individual profit maximization of a source i becomes:

$$\begin{aligned} &\max_{x_i} \Big[U_i(x_i) - q_i x_i \Big], \\ &x_i \geq 0. \end{aligned} \tag{6.7}$$

By solving Eqn. (6.7), each user obtains the optimal transmission rate x_i and achieves maximum profit. All sources determine their transmission rates in an adaptive manner as per the network load in each iteration. The individual optimal rates thus obtained may not lead to a social optimal solution (Eqn. (6.4)) for a general price vector $\mu = \{\mu_l\},\ l \in L$. Instead, [33]

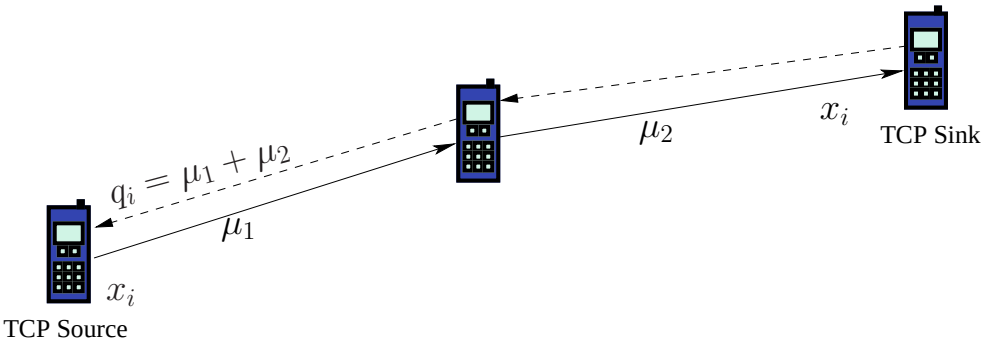


Figure 6.1: Feed-back Mechanism in Optimization Flow Control

provides an iterative method to solve the individual optimization, in which μ_l is considered as the Lagrangian. The solution converges to an optimal transmission rate ${x_i}^*$ for an optimal price ${\mu_l}^*$ of the links and maximizes Eqn. (6.4).

6.3 Congestion Control for Wireless Networks

The congestion control/avoidance techniques of TCP are proven to be effective in the wired network of the Internet. However, a direct extension of the congestion control of wired networks to wireless networks is not feasible [96–98]. In [97] the authors have presented a counter example illustrating in-feasibility of a direct extension. The basic reasons because of which congestion control techniques of wired network cannot be used in wireless network are as follows:

- The wireless network is broadcasting in nature. Therefore, collision of data as well as ACK packets can lead to congestion in the TCP layer. Moreover, because of the MAC layer functions, TCP timeouts can occur. Since TCP timeout is used as an indication of congestion, false congestion indications can result due to MAC layer functionalities.

- In addition to packet loss due to congestion, packets can also be lost due to the time varying nature of wireless channel. Distinguishing the packet loss due to congestion and the time varying nature of wireless channel is not trivial.

- Interference from other nodes also plays a significant role. Packet drops can occur because of interference also.

- The link capacities in a wireless network are not fixed, instead they depend upon the Signal to Interference and Noise Ratio ($SINR$) of the links. Since $SINR$ of a link depends

upon the power transmission policy of the network, packet loss at a link is affected by the power transmission policy of the network

Hence, applying the congestion control of wired networks directly in wireless networks may underestimate the capacity of the networks, leading to non-optimal performance and underutilization of resources.

There are various methods that have been proposed to solve the congestion control problem in wireless networks. These can be broadly divided into two categories: (1) Modification of TCP congestion control mechanism, taking the nature of wireless channel and network model into consideration; and (2) Changing the PHY and MAC layer functionalities based on the TCP's requirement. In the former approach, various new TCP congestion control techniques such as Split TCP [99], Ad-hoc TCP (ATCP) [100], TCP-BuS [101] and TCP-snoop [102] have been proposed in the literature. In the later approach, there are also few proposals, such as joint congestion and power control in wireless networks [30], joint scheduling and congestion control [103] and joint routing, scheduling and congestion control [104] that have been proposed in the literature. In this thesis, we concentrate on the second approach, i.e., we propose methods to improve TCP performances by exploiting PHY and MAC layer functionalities, discussed in detail in the next chapter.

As suggested before, there is a need of modification of the utility-based optimization framework for congestion control to accommodate the time varying channel and interference of neighboring nodes. Moreover, in a multi-hop wireless network, battery power of wireless nodes should also be considered in the optimization process, which is not addressed in any of these papers so far. We discuss these issues in the following sections.

6.3.1 Towards an Optimization Framework for Congestion Control in Wireless Networks

Various authors have studied the congestion control techniques of TCP in a wireless network. [30–32] have used power control along with congestion control in a wireless network and have analyzed it for TCP Vegas. In [105], the author has modeled the joint power control and utility maximization of a wireless network as a sum product problem. This approach is used to design a new Joint Optimal Congestion control and Power control (JOCP) algorithm (for a wireless network). [105] proposes a distributed power control algorithm which works jointly with

the existing TCP congestion control algorithm in order to increase the end-to-end throughput and energy efficiency of a multi-hop wireless network. This JOCP algorithm does not change the original TCP and layered structure of the network. Instead, it is a cross-layer approach, where the interactions among the Physical and Transport layer is used to increase the end-to-end throughput.

6.4 Cross-layer Congestion Control in Wireless Networks

In this section, we discuss a cross-layer congestion control technique based on sum product algorithms of [105] for TCP NewReno in a wireless network. Since in wireless networks, multiple packets can drop due to congestion as well as due to the time varying nature of the channel in one RTT, use of TCP NewReno is more appropriate. This is because, the value of $cwnd$ does not get decremented for more than half of its current value if multiple packet of a single window of data are lost in one RTT, which is common in wireless networks. Therefore, in TCP NewReno, packet loss due to wireless channel is also being taken care of. We consider the following steps to achieve a cross-layer congestion control algorithm for TCP NewReno in this thesis.

1. Formulation of TCP congestion control in terms of control system equations.

2. Use of optimization techniques to determine the maximum aggregate utility of sources, subject to capacity constraints and maximum transmission power [91] of wireless nodes.

6.4.1 System Model

The system model we use in this section is an extension of the system model illustrated in Section 6.2. We consider a network of N communicating nodes connected by L communicating links that are shared by I source-sink pairs. We illustrate this with an example through Figure 6.2. It is a small topology with six nodes, five links and two pairs of source-sink pairs or flows (between node 1 and node 5 and node 2 and node 6). Routing matrix H consists of the routing elements of the network.

For analyzing TCP congestion control, we consider a feedback system model which consists of a Source Controller and a Link Controller as shown in Figure 6.3. The input to the source

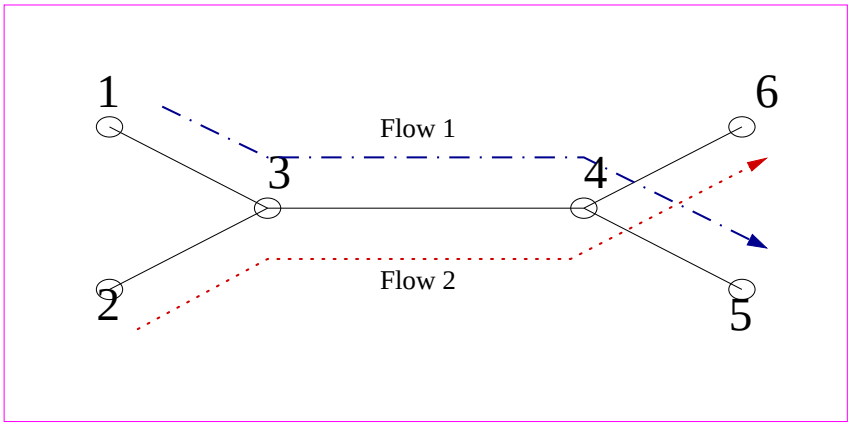


Figure 6.2: System Model/Topology

controller is the change in cost (Δq) of all I source-sink pairs and the output of the source controller is the change in rate of transmission (Δx) of all source-sink pairs. This is equivalent to solving the $USER(U_i(x_i), q_i)$ in Section 6.2.2. Similarly, the input to the link controller is the change in aggregate traffic in each link (Δy) and the output of the link controller is the individual link cost μ_l of each link l. This is equivalent to solving $NETWORK(H, C; q)$ in Section 6.2.2. The nodal points F_1 and F_2 determine the change in aggregate traffic in each link and aggregate price of each source-sink pair respectively. Though, there are time delays associated with the function of each block of the source-sink controller, we do not reflect that in our analysis for simplification. However, we incorporate the total delay as the Round Trip Delay, i.e., $RTT_i = RTT_i^f + RTT_i^b$, where RTT_i^f is forward delay (in practice the delay of a data packet from the source to sink) and RTT_i^b is the backward delay (in practice the delay of an ACK packet from the sink to source).

6.4.2 Problem Formulation

We formulate our problem statement initially for a wired network, similar to [106, 107] and then extend it to a wireless network. Our system model is based on fluid-flow abstraction. We consider long TCP connections instead of short ones in our model, as the former corresponds to majority of the TCP traffic in the Internet. We use control system equations [107] to determine the equilibrium point (x^*, y^*, q^*) for our system with the following assumptions: (1) resource allocation is characterized by a demand curve $x_i^* = f_i(q_i^*)$, (2) aggregate flow is less than or equal to the capacity of the link, i.e., $y_l^* \leq c_l$, and (3) the equilibrium queues are either empty

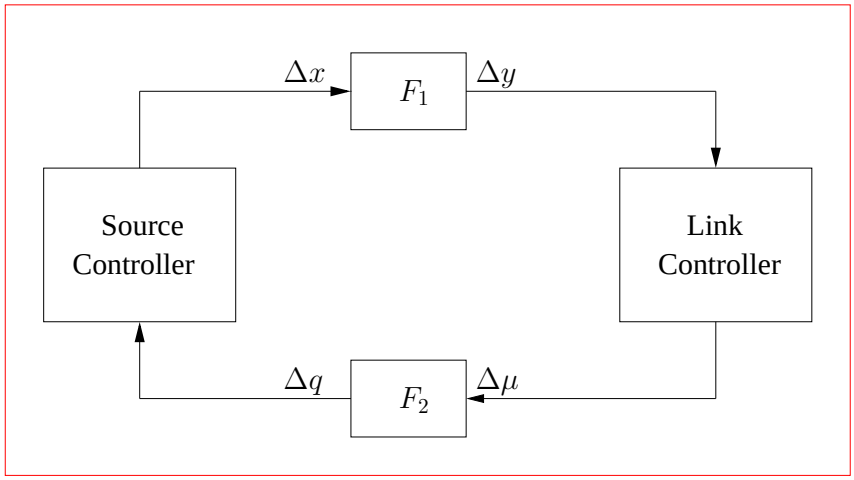


Figure 6.3: Feedback System Model

or small enough to avoid the queuing delays. The demand curve specified above is a decreasing function of price, which is equivalent to the use of an utility function $U_i(x_i)$ defined by [94], which is concave and continuously differentiable function of x_i for $x_i \geq 0$, i.e. $f_i = (U_i^{'})^{-1}$. At equilibrium, a source chooses its maximum profit by solving it as:

$$\max_{x_i^*} \left[U_i(x_i^*) - q_i^* x_i^* \right], \tag{6.8}$$

where $q^* = H^T \mu^*$, and x^*, y^*, q^* are the optimal fixed parameters. Since at equilibrium, each source attempts to maximize its profit (individual optimality) by choosing an appropriate rate; the individual optimality of Eqn. (6.8) can be re-written as a social optimality equation [78]:

$$\begin{aligned} &\max_{X \geq 0} \sum_i U_i(x_i), \\ &\text{s. t.,} \quad HX \leq C; \\ X = \{x_i\} \quad &\text{and} \quad C = \{c_l\} \end{aligned} \tag{6.9}$$

Using Karush-Kuhn-Tucker (KKT) condition on the social optimal equation (Eqn. (6.9)) as in [106], the constrained global maxima point[2] $X^* = \{x_i^*\}$ is expressed as:

$$\nabla \sum_i U(x_i) - \sum_l \mu_l \nabla (HX) = 0; \quad \forall \mu_l > 0, \tag{6.10}$$

[2]Since the utility function $U(x_i)$ is concave and increasing, KKT conditions are necessary and sufficient for a global maximum.

where μ_l is the Lagrangian Multiplier and is used as link prices (shadow prices) as discussed in [91] and [108]. Hence, Eqn. (6.10) is reduced to $U_i^{'}(x_i) = q_i^*$ at the global maximum point. The primal-dual distributed algorithm of the maximization equation Eqn. (6.9) signifies that the price is updated as a congestion parameter and is a dual variable. We now extend the optimization based congestion control technique to wireless network in the following section.

6.4.3 Extension to Wireless Networks

As discussed in [30], price is considered as a function of transmit power. By appropriate power control of all nodes, the maximum aggregate throughput can be attained. However, the increase of power in one node has a direct effect on the data rates of other nodes due to interference. Hence, we modify the maximization equation (Eqn. (6.9)) as follows:

$$\begin{aligned} & \max_{X \geq 0} \sum_i U_i(x_i), \\ \text{s.t.}, \quad & HX \leq C(P); \quad P = \{P_l\}, \\ & P_l \leq P_{l_{Max}}, \quad \forall l, \\ & P, X \geq 0, \end{aligned} \tag{6.11}$$

where P_l is the transmission power of a node in l^{th} link. Here, the link capacity c_l is a function of transmission power P_l in that link. Since it is observed in [109, 110] that power control significantly increases throughput in both CDMA and non-CDMA wireless networks, we use power control mechanism to achieve congestion control by the above optimization techniques. This is a cross-layer power and congestion control technique.

As discussed before, different variants of TCP have different utility functions. Since we use TCP NewReno, we now study the utility function of TCP NewReno and its properties.

6.4.4 Utility Function of a Practical Congestion Control Algorithm

In TCP NewReno, $cwnd$ is decreased by half for one or more congestion notification or triple dupacks or incremented by one for no mark in one RTT. We assume the marking probability to be a measure of congestion [106]. If τ_i is the equilibrium RTT (assumed to be constant for all), then transmission rate $x_i(t)$ of a source-sink pair $i \in I$ is expressed as:

$$x_i(t) = \frac{cwnd_i(t)}{\tau_i}, \tag{6.12}$$

where $cwnd_i(t)$ is the window size at time instant t. The transmission rate $x_i(t)$ is interpreted as the average rate over t. If there is no mark in one RTT (τ_i), then, $cwnd_i(t+\tau_i) = cwnd_i(t)+1$, else, $cwnd_i(t+\tau_i) = cwnd_i(t)/2$, for one or more mark in one RTT. Hence, the increase in window size is $1/\tau_i$ with probability $(1-\hat{p}_i(t))$ and the decrease is by $\frac{2cwnd_i(t)}{3\tau_i}$[3] with probability $\hat{p}_i(t)$. $\hat{p}_i(t)$ is the end-to-end probability that at least one packet is marked out of all packets of an window, in a period t for the source-sink pair i (the packet marking are independent). We approximate [106] $\hat{p}_i(t)$ for small $q_i(t)$ (aggregate price of all links in the route Eqn. (6.3)) as follows:

$$\hat{p}_i(t) = 1 - (1 - q_i(t))^{cwnd_i(t)} \approx cwnd_i(t)q_i(t) \tag{6.13}$$

Thus, the average change in window size[4] in the period t is:

$$x_i(t) = \frac{1}{\tau_i}\left(1 - \hat{p}_i(t)\right) - \frac{2}{3}\frac{cwnd_i(t)}{\tau_i}\hat{p}_i(t) \tag{6.14}$$

From Eqn. (6.13) and Eqn. (6.14), we determine the equilibrium parameters as follows:

$$\dot{x} = \frac{x_i(t)}{\tau_i} = \frac{1 - cwnd_i(t)q_i(t)}{\tau_i^2} - \frac{2}{3\tau_i}q_i(t)x_i(t)cwnd_i(t) \tag{6.15}$$

At equilibrium, setting $\dot{x} = 0$, $x_i(t) = x_i^*$ and $q_i(t) = q_i^*$ in Eqn. (6.15), we determine q_i^*:

$$q_i^* = \frac{3}{x_i(t)\tau_i(3 + 2x_i(t)\tau_i)}. \tag{6.16}$$

Now, using the fact that $U_i^{'}(x_i) = q_i^*$, the utility function is expressed as:

$$U_i(x_i) = \int q_i^* dx_i = \frac{1}{\tau_i}\log\left[\frac{x_i\tau_i}{2x_i\tau_i + 3}\right]. \tag{6.17}$$

Since the utility function of TCP NewReno, derived above, is concave for $x_i, \tau_i \geq 0$, our problem formulation in Eqn. (6.8) holds good.

[3]For a single NewReno flow, the window oscillates between $\frac{4cwnd_i(t)}{3\tau_i}$ and $\frac{2cwnd_i(t)}{3\tau_i}$ with an average of $cwnd_i(t)$ [106].

[4]The change in window size due to time-out is neglected here.

6.5 Discussions on Open Problems

In the recent years, there has been an explosion of research in cross-layer congestion control techniques with the prime motive of higher TCP throughput. There are some attempts, which have addressed on a cross-layer congestion and power control in wireless networks. However, practical implementations of cross-layer congestion and power control in real life network has not been explored properly. In addition, life time of the network, energy constraints of the wireless nodes and energy cost of the wireless nodes have also not been considered fully. The implementation methodologies, convergence analysis and effect of short flows on cross-layer congestion control and power control have also not been discussed in details [30–32].

In this direction, we attempt to solve some of these problems in this thesis. We attempt to propose a channel aware congestion and power control technique for a CDMA ad-hoc network, in which both congestion cost and energy cost are considered. We aim to minimize the energy spent for transmission and thereby increase the life-time of the network. Using, exhaustive simulations we show improvements in TCP throughput and convergence of the proposed algorithm. We also suggest implementation methodologies using ECN framework. We discuss the details of these problems in the next chapter.

Chapter 7

Joint Congestion and Power Control in CDMA Ad-hoc Networks

In the previous chapter, we have modelled TCP congestion control in wireless ad-hoc networks as a social optimization problem and employ congestion in a link as a cost function. Since the wireless nodes are mostly battery powered, the transmission power in a link should also be considered as a cost function. However, inclusion of cost function for transmission power in a link may lead to convergence problem. Therefore, further investigation on cross-layer based congestion and power control in ad-hoc networks is required. In addition, implementation of the cross-layer based congestion control in the protocol stack requires further investigation. To address the above mentioned issues, in this chapter, we consider CDMA ad-hoc networks as an example and propose a joint congestion and power control scheme in ad-hoc networks, which employs both congestion and energy cost. We model TCP congestion control in wireless networks as a social optimization problem and decompose the objective function defined for social optimization in wireless ad-hoc networks into two separate objective functions [32]. We solve these separate objective functions iteratively for the equilibrium transmission rates and link costs (both congestion cost and energy cost). The solution of the decomposed objective functions leads to a cross-layer approach involving TCP and the PHY layer, in which power transmission in a link is determined based on the interference and congestion in the network and the rate of transmission is determined based on congestion in the network. Note that we do not modify TCP congestion control protocol, instead we control the transmission power in an optimal manner to achieve higher throughput. We also propose an implementation method for

the cross-layer congestion control scheme using Explicit Congestion Notification (ECN) [34].

This chapter is organized as follows. In Section 7.1, we formulate a joint congestion and power control problem for a CDMA ad-hoc network. In Section 7.2, we discuss congestion cost, energy cost and attempt to solve the optimization problem in an iterative way. To determine the efficiency of our framework, we describe the experimental evaluation and discuss the results of our simulations in Section 7.3. In Section 7.3, we also investigate the convergence nature of the proposed scheme analytically and through simulations. We describe an implementation method of the cross-layer congestion control scheme using Explicit Congestion Notification (ECN) in Section 7.5.

7.1 System Model and Problem Formulation

The system model we use in this section is an extension of the system model illustrated in Section 6.2. We consider a CDMA ad-hoc network of N communicating nodes connected by L unidirectional communicating links and shared by I source-sink pairs. In the CDMA ad-hoc network every participating node has the capability to transmit at different power levels as per its requirements and transmits with different CDMA codes. Let each link be indexed by l and let $c_l(P_l)$ denote the capacity of link $l \in L$, where P_l is the transmission power on the link l. Note that the capacity of each link defined in Section 6.2 is fixed, whereas it is time varying in this section. Let $SINR_l$ be the Signal-to-Interference-and-Noise Ratio of link $l \in L$.

$$SINR_l = \frac{P_l G_{ll}}{\sum_{k \neq l} P_k G_{lk} + n_l}, \tag{7.1}$$

where G_{ll} is the path gain from the transmitter of link l to the receiver of the link l and G_{lk} is the path gain from the transmitter on link k to the receiver on link l. n_l is the thermal noise on the link l. We can determine the maximum capacity attainable in link l as:

$$c_l(P_l) = B \times \log(1 + MI \times SINR_l) \qquad \text{packets/sec}, \tag{7.2}$$

where B is the channel bandwidth and MI is a constant that depends on the modulation scheme used by the node for a successful transmission.

Let each source-sink pair $i \in I$ be associated with a transmission rate x_i and $U_i(x_i)$ be the utility associated with the transmisson rate x_i. Let the routing matrix H consist of the routing

elements of the network. The elements of the routing matrix are either "1" or "0", as explained in Eqn. (6.1). The aggregate flow y_l (packets/sec) at link l is defined in Eqn. (6.2) and the total cost q_i associated with a source-sink pair i is defined in Eqn. (6.3). We now attempt to solve the social optimization problem described in Eqn. (6.11), which we reproduce below (Eqn. (7.3)) for completeness.

$$\begin{aligned} & \max_{X \geq 0} \sum_i U_i(x_i), \\ \text{s.t.,} \quad & HX \leq C(P); \quad P = \{P_l\}, \\ & P_l \leq P_{l_{Max}}, \quad \forall l, \\ & P, X \geq 0, \end{aligned} \tag{7.3}$$

where $P_{l_{Max}}$ is the maximum power that a node can transmit. Since the utility function is concave and double differentiable and the constraints are linear, we can associate a Lagrangian Multiplier μ_l with the first constraint of Eqn. (7.3). The Lagrangian Multiplier μ_l can also be interpreted as the network cost or shadow price of transmission in the link. We now use KKT [111] optimality conditions to determine the stationary points ($X^* = \{x_i^*\}$ and $\mu^* = \{\mu_l^*\}$) by solving the complementary slackness conditions at equilibrium, i.e.,

$$\phi_{system}(X, P, \mu) = \sum_i U_i(x_i) - \sum_l \mu_l \big(\sum_i H_{li} x_i - c_l(P) \big). \tag{7.4}$$

Maximization of ϕ_{system} in Eqn. (7.4) is decomposed[1] as in [30]:

$$\begin{aligned} \max \quad & I(X, \mu) = \sum_i U_i(x_i) - \sum_l \mu_l \sum_i H_{li} x_i, \\ & \max \quad I(P, \mu) = \sum_l \mu_l c_l(P), \\ & \text{s.t.,} \quad X \geq 0; \quad 0 \leq P_l \leq P_{l_{Max}}. \end{aligned} \tag{7.5}$$

Both $I(X, \mu)$ and $I(P, \mu)$ are related by a common variable μ, which plays a significant role in determining the equilibrium window size (hence the data rate) and transmission power. Any change in μ results in change in throughput and transmission power. The first maximization

[1]Distributed solution is possible as long as there is an interaction between the two decomposed equations through some information passing. This is known as sum product algorithm [105].

equation involving $I(X,\mu)$ in Eqn. (7.5) is a direct consequence of congestion control of TCP. It is solved by the congestion control mechanism of TCP by increasing/decreasing the $cwnd$ size (and hence the individual data rates) for each flow in every RTT. Since the utility function of TCP NewReno $\left(U_i(x_i) = \frac{1}{\tau_i}\log\left[\frac{x_i\tau_i}{2x_i\tau_i+3}\right]\right)$ is concave and twice differentiable for $x_i, \tau_i \geq 0$, $I(X,\mu)$ will converge to a global maximum.

The second maximization equation involving $I(P,\mu)$ in Eqn. (7.5) is solved by choosing appropriate transmission power of wireless nodes. Before investigating the nature of $I(P,\mu)$, we discuss the nature of shadow price μ_l of a link. Since we consider both congestion cost and the cost of transmission, i.e., energy cost, the shadow price μ_l (and the Lagrangian Multiplier) can be expressed as:

$$\begin{aligned} \mu_l &= \lambda_l, \text{if "energy cost" is zero,} \\ &= \lambda_l + b_l, \text{otherwise,} \end{aligned} \tag{7.6}$$

where λ_l is the "congestion cost" and b_l is the "energy cost" of the link l. Before solving $I(P,\mu)$, we investigate the properties of the congestion cost and energy cost in the following section.

7.2 Solution Methodologies

As discussed earlier, we need to solve the sub-optimization problem $I(P,\mu)$ of Eqn. (7.5), such that the social optimization defined in Eqn. (7.3) is solved. In the following sections, we investigate the properties of both congestion cost and energy cost in detail.

7.2.1 Congestion Cost in TCP NewReno

We consider λ_l as the congestion cost of a link l. Since packet drop is used as an indication of congestion in TCP NewReno, probability of packet drop is used as congestion cost[2]. In this section, we derive a closed form expression for the congestion cost of a link in TCP NewReno.

We assume that the packets traversing in link l gets dropped if the aggregate traffic is more than the link capacity c_l. The probability of packet drop in a link can be modeled as packet loss probability in a $M/M/1/Z$ queue [90], where, Z is the buffer size at the link. In this kind of

[2]Concept of congestion cost is different for different versions of TCP. In TCP Vegas, queuing delay is used as an indication of congestion and hence the congestion cost.

queuing model, y_l and c_l serves as average arrival rate and average service rate [3] respectively. The loss probability (p_l) for this kind of queue at equilibrium is defined as [112]:

$$p_l(c_l, y_l) = \frac{(1-\rho)\rho^Z}{(1-\rho^{Z+1})}, \tag{7.7}$$

where $\rho := \frac{y_l}{c_l}$. By scaling the arrival rate, departure rate and buffer capacity by a factor K as in a many-source large-deviation scaling and by taking $K \to \infty$, we determine the loss probability as:

$$\begin{aligned} \lim_{K\to\infty} p_l(c_l, y_l) &= \lim_{K\to\infty} \frac{(1-\rho)\rho^{KZ}}{1-\rho^{KZ+1}} \\ &= \lim_{K\to\infty} \left[1 - \frac{1-\rho^{KZ}}{1-\rho^{KZ+1}}\right] \\ &= \frac{(y_l - c_l)^+}{y_l} = \frac{\max\big(0, (y_l - c_l)\big)}{y_l}. \end{aligned} \tag{7.8}$$

The above equation is valid only for $y_l > 0$. When $y_l = 0$, $p_l(c_l, y_l) = 0$. Since the loss probability (p_l) in TCP NewReno is considered as the shadow price/cost, congestion cost λ_l is expressed as:

$$\lambda_l(t) = p_l(c_l(t), y_l(t)) = \begin{cases} \frac{\max\left(0, (y_l(t) - c_l(t))\right)}{y_l(t)} & \text{if } y_l(t) > 0, \\ 0 & \text{if } y_l(t) = 0. \end{cases} \tag{7.9}$$

Since wireless channel as well as aggregate traffic in a link is time varying, we have introduced the time instant "t" in the above equation, i.e., the congestion cost is also time varying.

7.2.2 Energy Cost With and Without Battery Life Time

The energy cost is the cost of power spent by a wireless node to transmit its own traffic or to rely (forward) others' traffic. Energy cost is important when the nodes are battery powered and maximum node life time is desired. The energy cost "b_l" is a function of transmission power P_l (i.e., energy consumption) of the node in that link.

[3] The maximum capacity of the link is c_l and hence the service rate.

For successfully delivering a packet there is a requirement of minimum Bit Error Rate (BER) at the receiver. The minimum BER can be translated to a minimum $SINR$ in the link. In wireless link, the $SINR$ in one link depends not only on the transmission power in the same link, but also on the transmission power of other links. To maintain minimum $SINR$ in a desired link, the transmitting node needs to determine the P_l required for that link (which is affected by other's transmission also). We now define a linear pricing equation for the energy cost. Similar to the congestion cost, the energy cost is also time varying. We define the energy cost as:

$$b_l(t) = \theta + \alpha_1 f_1(\alpha_2 P_l(t)), \tag{7.10}$$

where θ is the minimum cost to keep the transmitter in "on" state (same as $P_{l_{Min}}$, the minimum power transmission) and α_1 and α_2 are constants which can be chosen for a pricing model by the network operator. The value of α_1 and α_2 also indicate whether energy cost or the congestion cost is dominant. The function f_1 is a piecewise monotonically increasing function. In Eqn. (7.10), we have not considered the battery life time/life time of the network, which is an important parameter. As the time progresses, the life time of a battery decreases, and hence choosing a path through this node becomes a costly affair. Therefore, the life time of the network should be reflected in the energy cost determination. We consider the energy cost as a function of amount of energy spent and the life time available for the battery of a particular wireless node. The energy cost on a path increases as the life time decreases. We consider a non-linear pricing model for the energy cost with battery life time as follows:

$$b_l(t) = \theta + \alpha_1 f_1(\alpha_2 P_l(t)) + \alpha_3 f_2(\alpha_4(P_{avl} - \sum_{t=0}^{t} P_l(t))) \tag{7.11}$$

The third sum term of Eqn. (7.11) decides the cost due to the amount of battery available in the wireless node. P_{avl} is the amount of energy available at the node before the start of the operation and α_3, α_4 are some constants. Similar to f_1, function f_2 can be any linear monotonic increasing function. The operator or the node can choose this function in an appropriate manner. In this chapter we use energy cost without battery life time for simplicity.

7.2.3 Nature of $I(P,\mu)$ and Solution to the Optimization Problem

With Congestion Cost only

We first consider the congestion cost only, i.e., $\mu_l(t) = \lambda_l(t)$ and solve $I(P,\mu)$ as follows. Without loss of generality, we assume that the bandwidth B and the modulation index MI of Eqn. (7.2) as unity and re-write $I(P,\mu)$ as follows:

$$\begin{aligned} I(P,\mu) &= I(P,\lambda) \\ &= \sum_l \lambda_l \log(SINR_l(P)). \end{aligned} \tag{7.12}$$

Since $I(P,\mu)$ is a concave function of a logarithmic transformed power vector, Eqn. (7.4) will converge to a global maxima resulting in optimum $X^* = \{x_i^*\}$, $\mu^* = \{\mu_l^*\}$. Solution of $I(P,\mu)$ can be obtained as follows:

$$\begin{aligned} I(P,\mu) &= \sum_l \mu_l \log\left(\frac{P_l G_{ll}}{\sum_{k\neq l} P_k G_{lk} + n_l}\right) \\ &= \sum_l \mu_l \left[\log(P_l) + \log(G_{ll}) - \log\left(\sum_{k\neq l} P_k G_{lk} + n_l\right)\right] \end{aligned} \tag{7.13}$$

Next, by differentiating $I(P,\mu)$ with respect to P_l, we evaluate the l^{th} component of the gradient $\nabla I(P,\mu)$.

$$\nabla I(P,\mu) = \frac{\mu_l(t)}{P_l} - \sum_{j\neq l} \frac{\mu_j(t) G_{jl}}{\sum_{k\neq j} P_k G_{jk} + n_j} \tag{7.14}$$

Now, we use the Steepest Descent method as in [111] to solve the maximization problem as:

$$\begin{aligned} P_l(t+1) &= P_l(t) + \delta\left(\nabla I(P,\mu)\right) \\ &= P_l(t) + \delta\left(\frac{\mu_l(t)}{P_l(t)} - \sum_{j\neq l} \frac{\mu_j(t) G_{jl}}{\sum_{k\neq j} P_k(t) G_{jk} + n_j}\right) \\ &= P_l(t) + \delta\frac{\mu_l(t)}{P_l(t)} - \delta \sum_{j\neq l} m_j(t) G_{jl}, \end{aligned} \tag{7.15}$$

where δ is a constant, called the step size in the direction of the gradient and $m_j(t) = \frac{\mu_j(t) SINR_j(t)}{P_j(t) G_{jj}}$. $m_j(t)$ can be interpreted as the interference received from node j.

In practice, power transmission $P_l(t)$ is always bounded by the minimum power level $P_{l_{Min}}$ and $P_{l_{Max}}$. Since the time scale of change of $cwnd$ is one RTT, the time scale of the change of transmission power P_l is also one RTT. From Eqn. (7.15), it is evident that the transmission power of a node in the next time instant $P_l(t+1)$ in a link l depends on three parameters, namely; (i) transmission power in the present time instant $P_l(t)$, (ii) shadow price $\mu(t)$, and (iii) the weighted sum of interference received from all neighboring nodes. The third factor is responsible for decreasing the transmission power of the concerned node in the next time instant. This is known as the co-operation principle in power control of wireless network. Increase in $\mu_l(t)$ is responsible for increasing power in the next time instant. Intuitively, more the shadow loss, more the congestion, thereby increasing the transmission power in the next time instant.

With Congestion Cost and Energy Cost

We now consider both congestion cost as well as energy cost (without battery life time) to solve $I(P,\mu)$. We re-write $I(P,\mu)$ with both congestion and energy cost as follows:

$$\begin{aligned} I(P,\mu) &= \sum_l \mu_l \log(SINR_l(P)) \\ &= \sum_l (\lambda_l + b_l)\log(SINR_l(P)). \end{aligned} \tag{7.16}$$

Instead of investigating the concave nature of $I(P,\mu)$, we transform the power vector P to a logarithmic power vector $\tilde{P}$ and then investigate the concave nature of the transformed $I(\tilde{P},\mu)$. Let $\tilde{P}_l = \log P_l, \forall l$. We write $I(\tilde{P},\mu)$ as follows:

$$\begin{aligned} I(\tilde{P},\mu) &= \sum_l \mu_l \log\left[\frac{G_{ll}e^{\tilde{P}_l}}{\sum_{k\neq l} G_{lk}e^{\tilde{P}_k} + n_l}\right] \\ &= \underbrace{\sum_l \lambda_l \left[\log(G_{ll}e^{\tilde{p}_l}) - \log\left(\sum_{k\neq l} e^{\tilde{P}_l + \log G_{lk}} + n_l\right)\right]}_{I_1(\tilde{P},\mu)} \\ &+ \underbrace{\sum_l b_l \left[\log(G_{ll}e^{\tilde{p}_l}) - \log\left(\sum_{k\neq l} e^{\tilde{P}_l + \log G_{lk}} + n_l\right)\right]}_{I_2(\tilde{P},\mu)} \\ &= I_1(\tilde{P},\mu) + I_2(\tilde{P},\mu) \end{aligned} \tag{7.17}$$

From Eqn. (7.17), it is evident that $I_1(\tilde{P}, \mu)$ is concave in $\tilde{P}$, as $\log(G_{ll}e^{\tilde{P}_l})$ is linear in $\tilde{P}$ and $\log\Big(\sum_k e^{\tilde{P}_l+\log G_{lk}} + n_l\Big)$ is convex in $\tilde{P}$.

$$\begin{aligned} I_2(\tilde{P}, \mu) &= \sum_l \theta\Big[\log(G_{ll}e^{\tilde{p}_l}) - \log\Big(\sum_{k\neq l} e^{\tilde{P}_l+\log G_{lk}} + n_l\Big)\Big] \\ &+ \sum_l \alpha_1 f_1(\alpha_2 P_l)\Big[\log(G_{ll}e^{\tilde{p}_l}) \\ &- \log\Big(\sum_{k\neq l} e^{\tilde{P}_l+\log G_{lk}} + n_l\Big)\Big] \end{aligned} \tag{7.18}$$

Similarly, from Eqn. (7.18), it is evident that $I_2(\tilde{P}, \mu)$ is concave as $f_1(\alpha_2 P_l)$ is linear and monotonic in P_l. Hence, $I(\tilde{P}, \mu)$ is concave in $\tilde{P}$ and in P. Since $I(X, \mu)$ and $I(P, \mu)$ are concave and double differentiable, Eqn. (7.4) will converge to a global maxima resulting in optimum $X^* = \{x_i^*\}$, $\mu^* = \{\mu_l^*\}$. The solution methodologies to find the global maxima are explained below.

Each source-sink pair i solves the first maximization equation involving $I(X, \mu)$ as it knows its own utility function $U_i(x_i)$ and the end-to-end cost (which is fed back by the system) by using TCP NewReno congestion control in every RTT. We solve the second maximization equation involving $I(P, \mu)$ by choosing appropriate transmission powers of the wireless nodes as discussed below. Differentiating $I(P, \mu)$ with respect to P_l, we evaluate the l^{th} component of the gradient $\nabla I(P, \mu)$. We use the Steepest Descent method to solve the maximization problem with a small step size δ to obtain the transmission power as:

$$P_l(t+1) = P_l(t) + \delta\Big(\nabla I(P, \mu)\Big). \tag{7.19}$$

As discussed earlier, the time scale of change of transmission power is one RTT. Each node determines its required transmission power P_l (in practice the transmission power of a node is bounded by minimum and maximum power levels) and network cost μ_l of each outgoing link associated with it and passes on these information to its neighbors, such that end-to-end cost of a path can be determined by each source-sink pair i. Each source-sink pair i determines its transmission rate using TCP NewReno and transmission power using Eqn. (7.19), which is equivalent to solving a joint congestion control and power control for an ad-hoc network. It is a channel aware congestion control technique. We suggest an algorithm to implement this joint congestion and power control in Algorithm 4. The parameter Δ in Algorithm 4 is a small

tunable system parameter. The value of Δ is set as the granularity of increase/decrease of power level of a node.

In power control techniques, each wireless node needs to advertise its $SINR_l$ requirement either on a separate channel [110] or on the same channel [113]. These nodes update their path gains, noise levels, interference caused by other nodes either after receiving the advertised signal or in a periodic manner.

Algorithm 4 :Channel aware Congestion Control for CDMA Ad-hoc Network with Congestion and Energy Cost

$cwnd_i(0) \leftarrow 3 \ \forall i$
$P_l(0) \leftarrow P_{l_{Max}} \ \forall l$
$c_l(0) \leftarrow c_{l_{Max}} \ \forall l$
$b_l(0) \leftarrow \theta \ \forall l$
$\lambda_l(0) \leftarrow 0 \ \forall l$
$\mu_l(0) \leftarrow \theta \ \forall l$
$t \leftarrow 1$
while TRUE **do**
  Update G_{ll} and G_{lk} periodically
  $SINR_l(t) \leftarrow \frac{P_l(t-1)G_{ll}}{\sum_{k \neq l} P_k(t-1)G_{lk}+n_l} \ \forall l$
  $P_l(t) \leftarrow P_l(t-1) + \delta(\nabla I(P,\mu)) \ \forall l$
  if $|P_l(t) - P_l(t-1)| \leq \Delta$ **then**
    $P_l(t) \leftarrow P_l(t-1) \ \forall l$
  else
    $P_l(t) \leftarrow \min(P_l(t), P_{l_{Max}}) \ \forall l$
  end if
  Update $cwnd_i(t), \tau_i$ from TCP Module $\forall i$
  $x_i(t) \leftarrow \frac{cwnd_i(t)}{\tau_i} \ \forall i$
  $y_l(t) \leftarrow \sum_i H_{li} x_i \ \forall l$
  $b_l(t) \leftarrow \theta + \alpha_1 f_1(\alpha_2 P_l(t)) \ \forall l$
  $c_l(t) \leftarrow \frac{1}{T} \log(1 + MI \times SINR_l(t)) \ \forall l$
  $\lambda_l(t) \leftarrow \frac{\max(0, y_l(t) - c_l(t))}{y_l(t)} \ \forall l$
  $\mu_l(t) \leftarrow \lambda_l(t) + b_l(t) \ \forall l$
  $q_i(t) \leftarrow \sum_l H_{li} \mu_l \ \forall i$
  $t \leftarrow t + 1$
end while

7.3 Experimental Evaluation of Channel aware Congestion Control Algorithm

In this section, we describe simulation experiments that have been performed to evaluate channel aware congestion control algorithm. All the simulations have been conducted using im-

plementations of cross-layer congestion control algorithm in MATLAB [68]. We consider a CDMA ad-hoc network with six wireless nodes and two pairs of TCP source-sink pairs (1-5) and (2-6) as in shown in Figure 7.1 (same topology is used here as shown Figure 6.2). All nodes in our simulation are capable of transmitting and receiving "network cost". We also consider that all six nodes are TCP NewReno agents. Depending upon the network costs, we update the transmission power of the participating nodes. We set the TCP retransmission timeout to be $4 \times RTT$. We update RTT by using the exponential averaging technique as: $RTT = \Upsilon RTT_{estimated} + (1 - \Upsilon)RTT_{measured}$. We assume that $\Upsilon = 0.85$ for our simulations. We assume fixed packet lengths of 1000 bits size. We assume that the time required for transmission in each of the segments (Figure 7.1) 1-3, 2-3, 3-4, 4-5 and 4-6 are same. We conduct three different sets of experiments. They are: (i) Channel aware Congestion control with Congestion Cost (CC-CC), (ii) Channel aware Congestion control with Network Cost (CC-NC) and (iii) congestion control Without Power Control (W-PC) (congestion control with fixed power transmissions)

The path loss exponent due to distance is set as $\gamma = 4$. We consider AWGN with PSD N_0 = 0.35 (4.5 dB/Hz). We also simulate shadowing in our experiments. The shadowing is modeled as Log-normal with mean zero and standard deviation (σ) 8 dB. In each simulation run, the channel gain due to Log-normal shadowing is kept fixed for the entire duration of simulation. We also repeat the experiments with different Log-normal shadowing with σ of 4, 6, 8, 10 and 12 dB. We assume channel reciprocity, i.e., both forward and reverse link gains are the same.

We implement TCP NewReno in MATLAB. We set $cwnd_{initial}$ = 3 packets, $P_{l_{Min}}$ = 3 mwatt and $P_{l_{Max}}$ = 10 mwatt in our simulations. We determine the data rate x_i by using the relation $x_i(t) = \frac{cwnd_i(t)}{\tau_i}$, whereas $cwnd_i(t)$ and τ_i are updated using TCP NewReno congestion control principle. We also consider different step size (δ = 0.1, 0.2 and 0.5) to verify the robustness and convergence of our algorithm. The system parameters used for simulations are presented in Table 7.1. The value of each parameter observed has been averaged over 20 independent simulation runs.

7.3.1 Simulation Results

We simulate TCP NewReno congestion control mechanism with and without power control techniques. In Figure 7.2 and 7.3, we plot the $cwnd$ variation of both flows using channel aware congestion control mechanism with congestion cost and network cost respectively. In

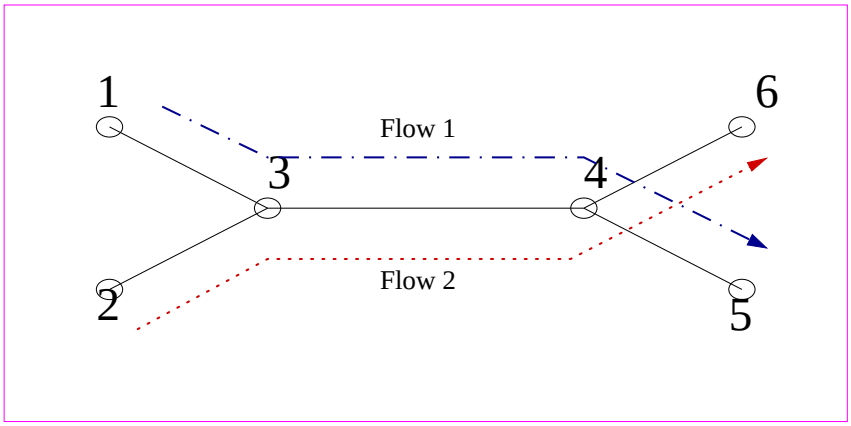


Figure 7.1: System Model/Topology

Table 7.1: Summary of Simulation Parameters in Channel aware Congestion Control

Simulation Parameter	**Value**
$cwnd_{initial}$	3 packets
$P_{l_{Min}}$	3 mwatt
$P_{l_{Max}}$	10 mwatt
Number of Flows	2
Number of Nodes	6
Υ	0.85
Packet Length	1000 bits
Step Size (δ)	0.1, 0.2, 0.5
Path Loss Exponent (γ)	4
Log-normal Shadowing (σ)	4, 6, 8, 10, 12 dB

Figure 7.4, we plot the $cwnd$ variation of both flows with fixed power transmission. From these figures, we observe that the average $cwnd$ size of channel aware congestion control scheme with congestion cost (CC-CC) is 8.17 packets and with network cost (CC-NC) is 6.5 packets, whereas it is 6.4 packets for the conventional congestion control without power control (W-PC) scheme. From this, we observe that there is a average $cwnd$ size gain of 27% of CC-PC over W-PC scheme and 1% of CC-NC over W-PC scheme. We also observe that even though the gain in average $cwnd$ for CC-NC over W-PC is not significant (1% gain), there is less fluctuation in the $cwnd$ evolution of CC-NC as compared to W-PC. This results in less re-transmission, resulting

in higher TCP throughput. This verifies that the channel aware congestion control scheme provides stabilized and higher throughput than the conventional congestion control scheme.

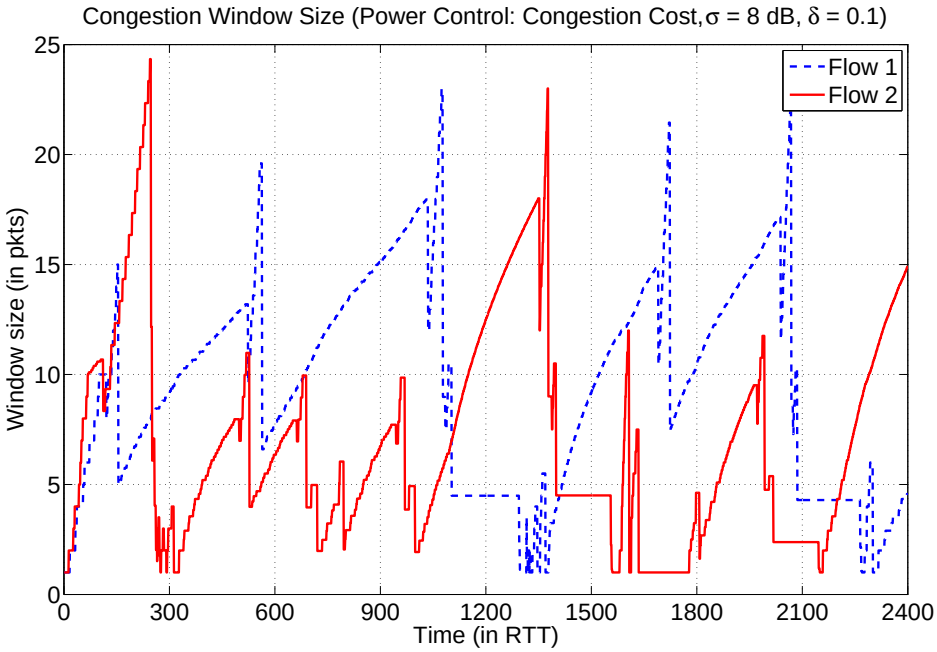


Figure 7.2: Variation of $cwnd$ Size with Power Control (Congestion Cost only)

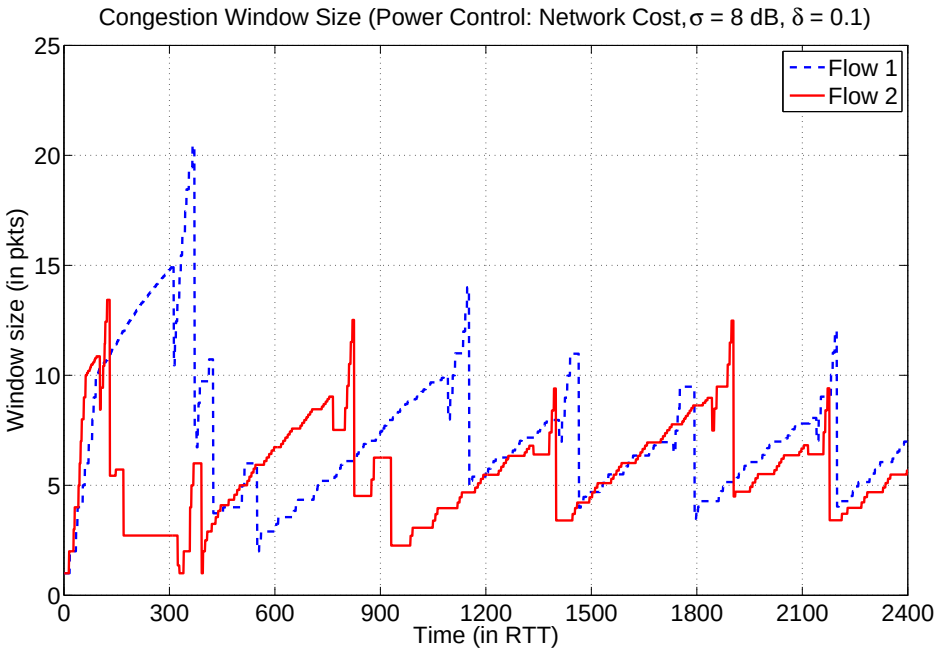


Figure 7.3: Variation of $cwnd$ Size with Power Control (with Network Cost)

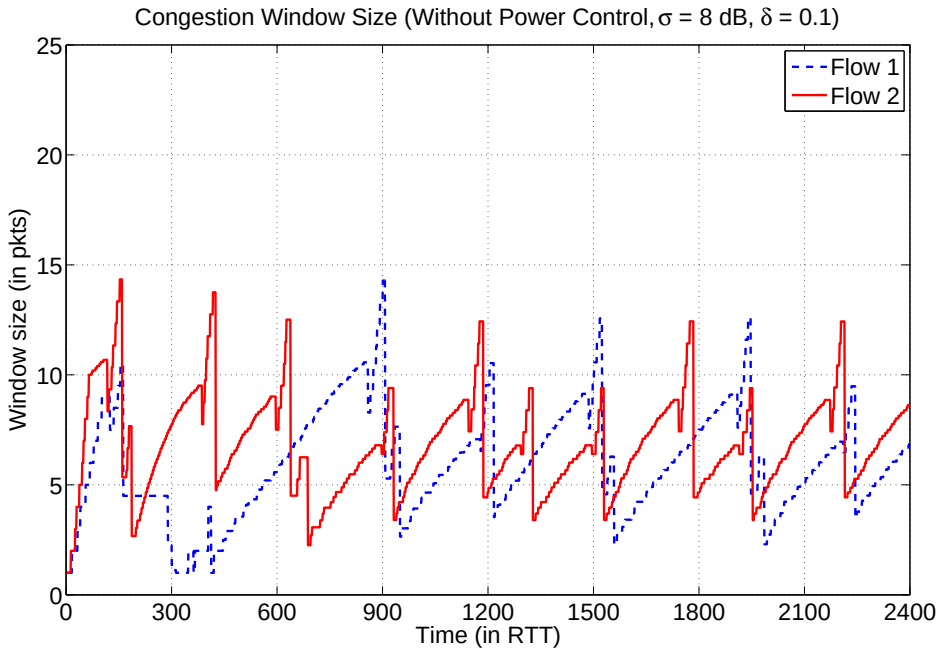


Figure 7.4: Variation of $cwnd$ Size without Power Control

Simulation with Different Log-normal Shadowing

We also conduct several experiments with different standard deviation (σ) of Log-normal shadowing. We plot the variation of shadow price for different standard deviation (σ) of Log-normal shadowing in Figure 7.5 and 7.6. From these figures, we observe that shadow price increases as the σ increases. We also observe the average transmission power of individual nodes at different channel states (cf. Figure 7.7). From Figure 7.7, we observe that the average transmission power decreases as the channel gain deteriorates (similar to Opportunistic scheduling). Moreover, we observe that for σ = 8 dB, nodes transmit in an average of $P_{l_{Avg}}$ = 9.33 mwatt (when we consider congestion cost only) and at $P_{l_{Avg}}$ = 5.67 mwatt (when we consider network cost), whereas nodes transmit at a maximum power level ($P_{l_{Mmax}}$ = 10 mwatt) in the conventional scheme. This verifies that cross-layer congestion control not only improves average throughput, but also saves transmission power (and hence the life time of the nodes) significantly.

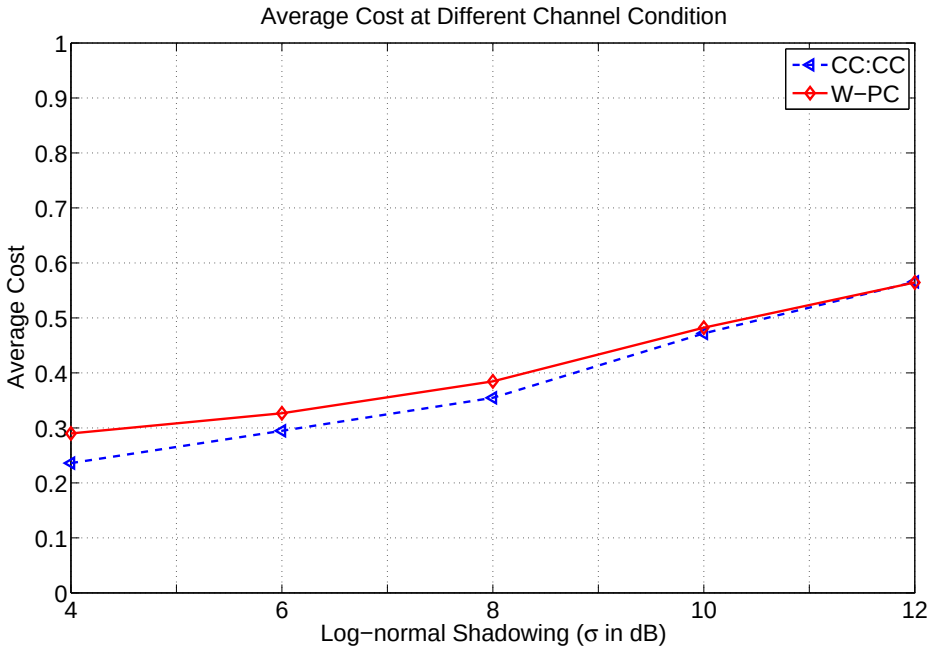


Figure 7.5: Variation of Congestion Cost at Different Channel Condition

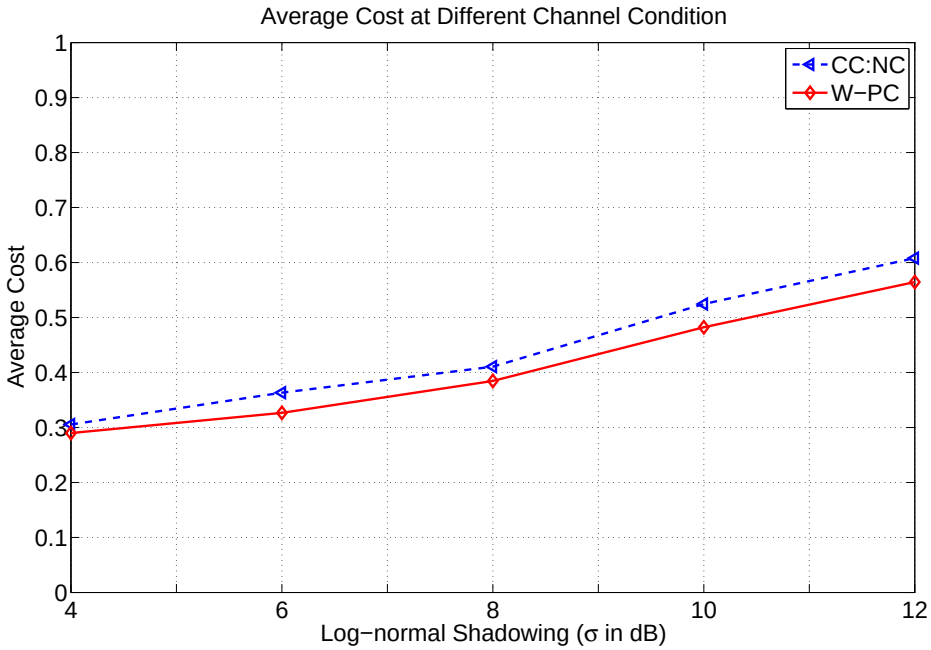


Figure 7.6: Variation of Network Cost at Different Channel Condition

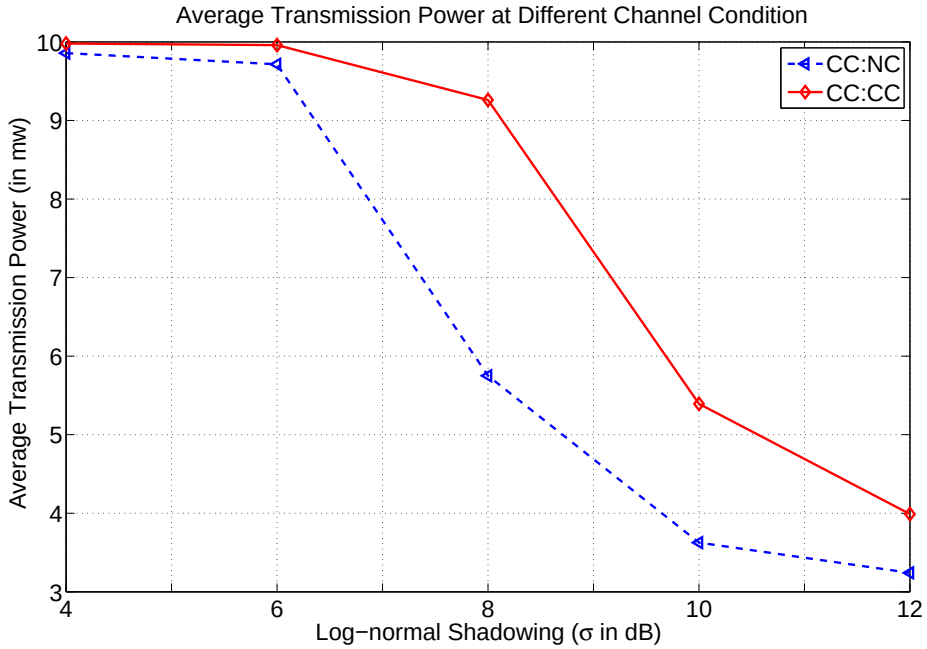


Figure 7.7: Average Transmission Power at Different Channel Condition

In Figure 7.8 and 7.9, we plot the average TCP throughput obtained out of various schemes over different σ of Log-normal shadowing. From these figures, we observe that even though cross-layer congestion control (for both congestion and network cost) provides higher throughput than the conventional congestion control scheme, the rate of decrease of throughput of cross-layer congestion control scheme is more than that of normal scheme (without power control) as the σ of Log-normal shadowing increases. The higher rate of throughput drop for the cross-layer schemes can be attributed to the fact that as the shadowing increases, transmission power decreases resulting in decrease in the average throughput.

7.4 Convergence Analysis of Channel aware Congestion Control Scheme

In TCP NewReno, we consider the probability of marking/drops or shadow price (μ_l) as a measure of congestion. We also observe that μ_l is controlled by the transmission power P_l as it (μ_l) is used as the common variable between the two sub optimization problems of Eqn. (7.5). Since power P_l is bounded by minimum ($P_{l_{Min}}$) and maximum ($P_{l_{Max}}$) power levels and the shadow price is zero at equilibrium state (total incoming traffic and capacity of a link are

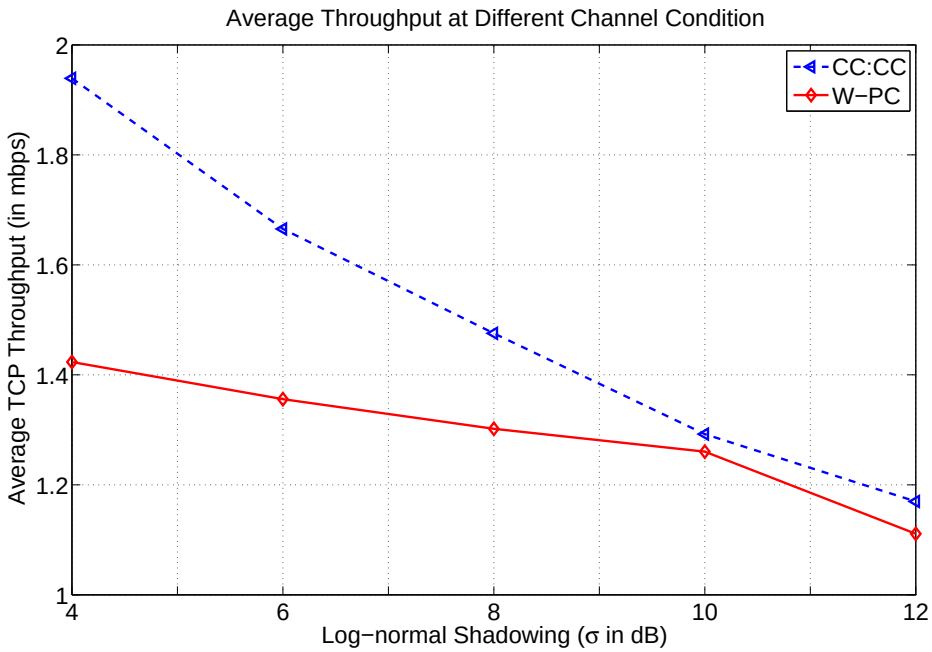


Figure 7.8: Average Throughput Achieved at Different Channel Condition (with Congestion Cost)

same, resulting in no packet loss), a complimentary slackness condition between the primal variable x_i and the dual variable μ_l of Eqn. (7.5) can be achieved. Now, going along the line of the proof given in [31] (see Theorem 1 of [31]), we can prove that the channel aware congestion control algorithm derived from Eqn. (7.5) will converge to a global optimum point. This requires that the minimum $SINR$ should be greater than one, else, $I(P,\mu)$ cannot be approximated as $I(P,\mu) = \sum_l \mu_l \log(SINR_l(P))$. Also, it is clear from our discussions in Section 7.2.3, that $I(P,\mu)$ is a strictly concave function of logarithm of power transmission vector. Hence, the Lagrangian Multiplier μ_l facilitates a global maximization of Eqn. (7.4) and ensures convergence. The step size (δ) of Steepest Descent method of optimization in Eqn. (7.15) decides the rate of convergence towards a global optimum point. The convergence is guaranteed as long as no new user enters or old users leave the network. For any addition and deletion of nodes/users, this algorithm will again take some iterations to converge.

To evaluate the convergence properties of the channel aware congestion control schemes, we perform simulations with three different step sizes (δ = 0.1, 0.2 and 0.5) and observe the number of iterations to converge (to an optimum transmission power value). We observe that the proposed algorithm converges for small step sizes (δ =0.1 and 0.2) and oscillates for large step size (δ = 0.5).

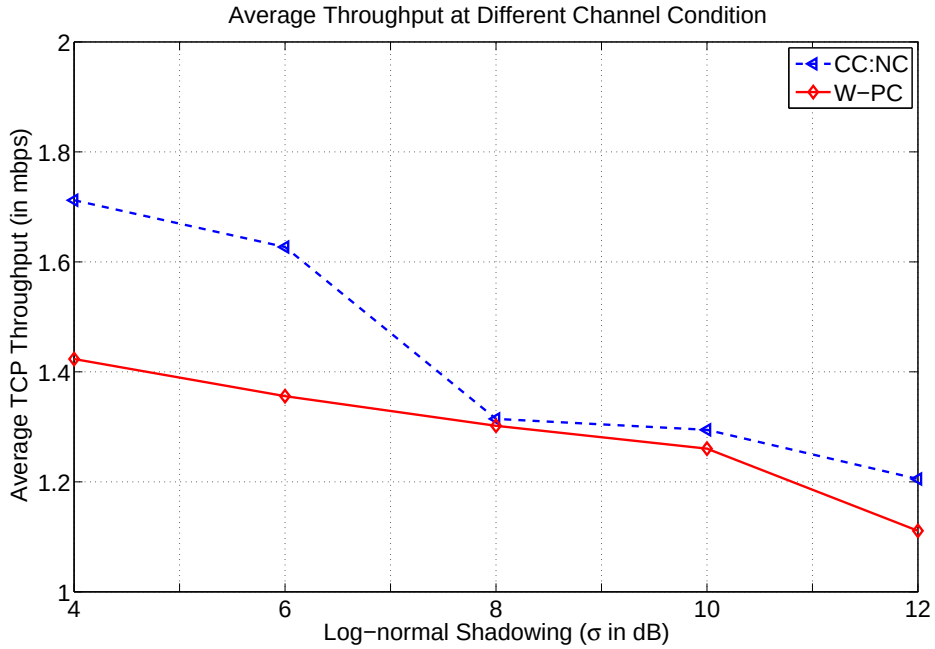


Figure 7.9: Average Throughput Achieved at Different Channel Condition (with Network Cost)

7.4.1 Convergence Analysis with Flow Alteration

To investigate the convergence of the proposed scheme with flow alteration (addition/deletion of flows), we select a different topology illustrated in Figure 7.10 and Figure 7.11. We assume δ = 0.1. We consider two cases, involving four flows and five links. In Case-I, we have four flows (Figure 7.10), and perform simulations to obtain the equilibrium rates of each flow, number of iterations it takes to converge, transmission power in each link and the total aggregate traffic in each link. We plot the variation of transmission rate vs. iteration number in Figure 7.12. From this figure, we observe that the algorithm takes around 150-180 iterations to converge to its equilibrium value. We also present the converged transmission powers and aggregate rates in Table 7.2 and rate of transmissions in Table 7.3. From these tables, we observe that in Case-I, L-3 and L-4 are most congested links as they accommodate three flows each. Hence as expected, the transmission powers in those links are higher as compared to other links. Similarly, we observe that L-1 is the least congested link and hence the transmission power is the least in L-1.

After reaching convergence, we delete Flow-4 involving L-5 and L-6 and create a new flow involving L-1 and L-2 as shown in Case-II. We observe that it takes another 180-200 iterations (Figure 7.12) to converge, and hence we claim that our algorithm converges to addition and/or deletion of flows in a realistic time frame. In Case-II, L-2 and L-3 are most congested as they

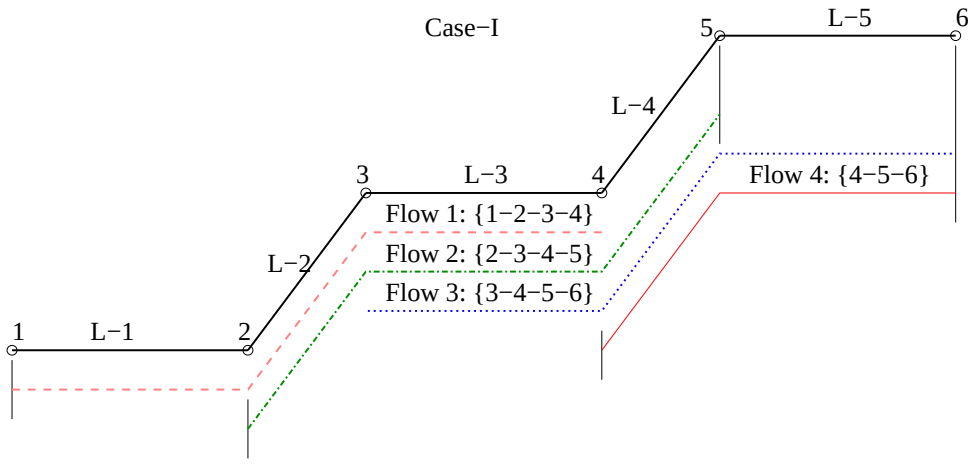


Figure 7.10: Topology for Convergence Analysis

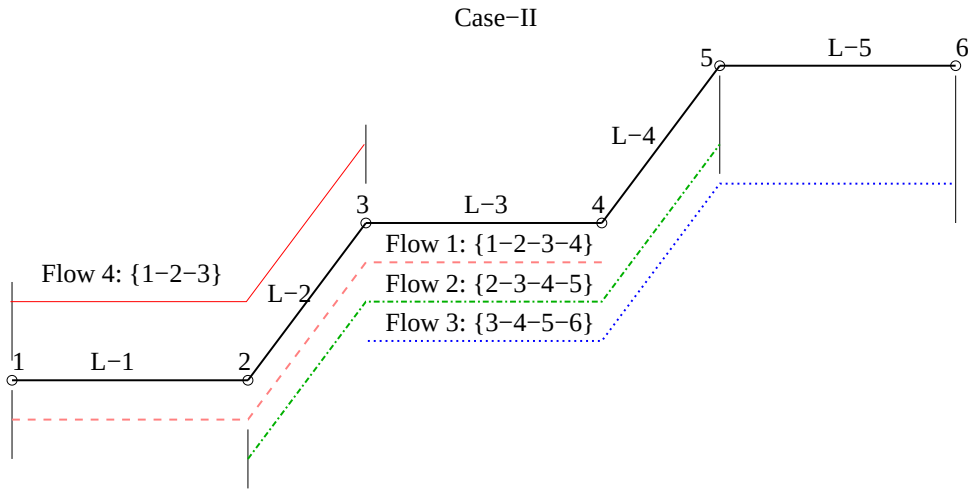


Figure 7.11: Topology for Convergence Analysis with Flow Alteration

accommodate three flows each and hence the transmission power in those links are high.

7.5 Implementation of Channel aware Congestion Control Using ECN Framework

In this section, we propose implementation of channel aware congestion control scheme in wireless networks using ECN framework [34]. We begin with the discussed on current implementation of ECN in a wired network and then extend that to an ad-hoc wireless network.

Table 7.2: Power Transmission and Link Usage

Links	P_l	y_l
L-1	5.2589	0.3147
L-2	8.479	0.5397
L-3	9.9616	0.8231
L-4	10.2413	0.9823
L-5	5.6831	0.7891

(a) Case - I

Links	P_l	y_l
L-1	6.9911	0.7128
L-2	13.6162	0.9489
L-3	12.6060	0.8493
L-4	10.0886	0.5816
L-5	7.6860	0.3453

(b) Case - II

Table 7.3: Flow Rate after Convergence

Flows	x_i
Flow 1	0.3147
Flow 2	0.2250
Flow 3	0.2835
Flow 4	0.5056

(a) Case - I

Flows	x_i
Flow 1	0.2677
Flow 2	0.2361
Flow 3	0.3455
Flow 4	0.4452

(b) Case - II

7.5.1 Explicit Congestion Notification (ECN) in Wired Network

As discussed in Section 6.1, Explicit Congestion Notification (ECN) uses marking of packets as a method of congestion notification in AQM. ECN can be used in conjunction with Random Early Detection (RED), which marks a packet instead of dropping it when the average queue size is between the limits min_{th} and max_{th}. Since ECN marks packets before the actual congestion, it is useful for protocols like TCP that are sensitive to even a single packet loss. Upon receipt of a marked packet (congestion) from the intermediate router/node, TCP sink informs the source (in the subsequent ACK) about incipient congestion which in turn triggers the congestion avoidance algorithm at the source. ECN requires support from both the router as well as the end nodes (source and the sink).

In the Internet Protocol (IP) header of TCP/IP packets, ECN field uses two bits, resulting in four ECN code points, i.e., $00, 01, 10$ and 11. The code points 10 and 01 are known as ECN-Capable Transport (ECT) code-points and are set by the source to indicate that the end

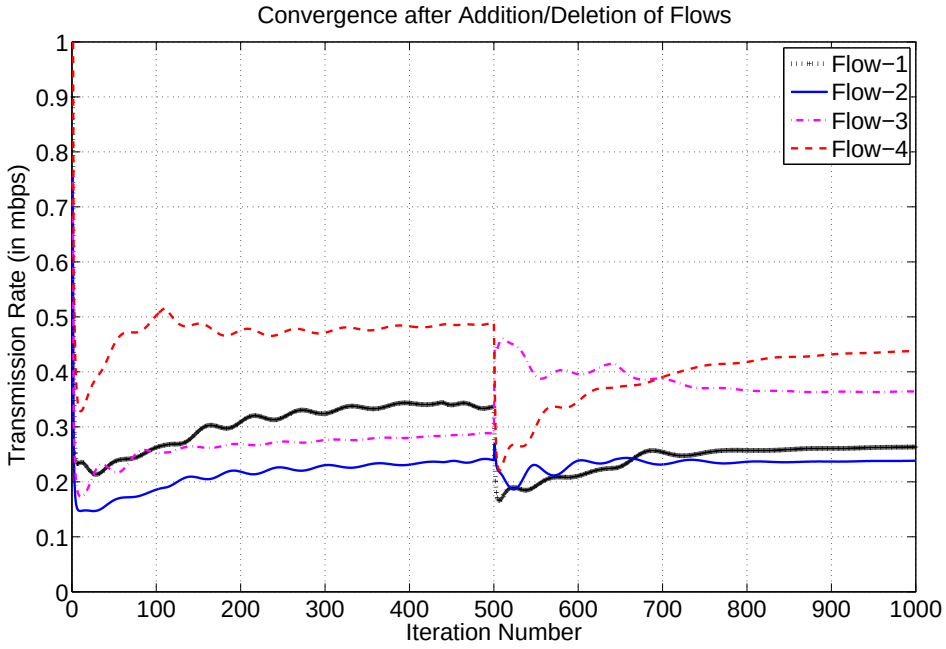


Figure 7.12: Convergence after Addition/Deletion of Flows

points are ECN capable. Code point 01 is known as $ECT(1)$, whereas code point 10 is called as $ECT(0)$. ECN code point 00 is used to indicate that the end node is not ECN compatible. Even though the end nodes are free to choose $ECT(0)$ or $ECT(1)$, $ECT(0)$ is employed in the current implementations. By employing $ECT(0)$, code point 01 is left unused. ECN employs ECT and CE flags in the IP header for signaling between routers and end points. In the TCP header, it uses ECN-Echo (ECE) and Congestion Window Reduced (CWR) flags for TCP end point signaling. Hence, the four flags (ECT, CE, ECE, CWR) are used by ECN for AQM in the network/routers. For a typical TCP/IP connection, ECN uses the following steps to control congestion.

- Source sets an ECT code point in the transmitted packet to indicate that it is ECN capable.

- When an ECN capable router experiences congestion by measuring the average queue size in RED or by some other mechanism, it sets the CE code point in the IP header and forwards the packet (ECT set packet, sent by the source) towards the sink.

- An ECN capable sink receives the packet (with ECT and CE code point set), and sets the ECE flag in the acknowledgment packet due for this packet to the source.

- The source receives the TCP ACK with ECE flag set, and reacts to the congestion as if a packet had been dropped.

- The source sets the CWR flag in the TCP header of the next packet sent to the sink to acknowledge its receipt of the ACK packet with ECE flag and its reaction to the ECE flag. The source drops the $cwnd$ as if there is a packet drop in the network.

7.5.2 Extension of ECN to Wireless Ad-hoc Networks

As discussed earlier in this thesis, the current TCP implementations employ $ECT(0)$ resulting in $ECT(1)$ being unused. We employ both $ECT(0)$ and $ECT(1)$ code points in our proposal. We call this as $ECT(X)$. By doing this, we finally get three code points instead of two (00 is used for non-ECN compatible end points) code points. We also assume that the wireless nodes are ECN capable. Hence, there is no need to transmit ECT code points separately to indicate that the node is ECN capable or not. However, for backward compatibility, we use code point 00 to indicate that the end node is not ECN compatible. The ECN routers (wireless routers in our case) use both ECT and CE for notifying congestion to the sink. This is different from the one bit ECN used in the wired network. The sink, after receiving the ECT and CE sets packets from the source through the router, acknowledges to the source by setting the ECE flag of the TCP header appropriately. However, this operation in our scheme needs two ECE flags instead of the regular one ECE flag of original ECN scheme. To achieve this, we suggest to use another unused bit from the four reserved bits of the TCP header. Therefore, the sink in our scheme uses two ECE bits instead of one ECE bit. It replicates the ECT and CE flags of the IP header to the two ECE flags of the TCP header. On receiving the ECE set packets, the source acknowledges by setting the CWR flag of the TCP header.

Before discussing the exact implementation of ECN in wireless ad-hoc networks, we define the following terms:

- **Short-term Congestion:** In wireless networks, packets get dropped either due to congestion or due to the time varying nature of wireless channel. If the packet drop probability due to channel fading (this can be determined from the average $SINR$ received at the node) is more than the probability of packet drop due to congestion (this can be determined from the average queue occupancy of the node), then this can be termed as

short-term congestion.

- **Long-term Congestion:** If the packet drop probability due to both congestion (buffer overflow) and due to channel fading are high, then we term this as *long-term congestion*.

As discussed earlier, our ECN scheme uses both ECT and CE flags for notifying the congestion to the sink, and old/new ECE flags for notifying the congestion to the source. If the intermediate router does not experience congestion, then it sets 01 to the ECN code points (ECT and CE flags). It sets the ECN code points to 10, if it experiences short-term congestion, whereas it sets to 11 if it experiences long term-congestion. On receiving the ACK packets with ECE flags set as 01 or 10 or 11, the source modifies the $cwnd$ as follows:

- If the ECE flags are set as 01, then there is no congestion in the network. Therefore, the source can increase the window size as per the version of TCP (Reno, NewReno, Tahoe etc.) it is using.

- To notify short-term congestion, we use 10 as the ECE flag. Upon receiving this from the sink, the source instead of modifying the $cwnd$ (by increasing or decreasing using AIMD) it continues to transmit at the current $cwnd$ size. This can help the network from the synchronization problem. Also, it helps the source nodes by transmitting at a higher rate, instead of decreasing the $cwnd$ size.

  The intuition behind this proposal is as follows: Since we are dealing with time varying wireless channel in the proposed scheme, there will be certain times, when the channel fading is high and there is no congestion in the network. Hence, packets get dropped by the channel itself. By using the original ECN with conventional TCP like TCP Reno, NewReno and Tahoe, the source will drop its $cwnd$ by half, resulting in degradation in the throughput. However, in the proposed scheme, $cwnd$ is not dropped by half, instead it is maintained at this value. This is due to the short-term congestion in the network. Packet drops due to fading can be avoided by the power control at the PHY layer.

- To notify long-term congestion in the network, we use ECE flag 11 in our scheme. On receiving ACK packets with 11 as the ECE flag, the source drops the $cwnd$ by half. This is equivalent to packet drops in the network, due to congestion as well as due to the channel state.

The intuition behind this proposal is as follows: In long-term congestion, the probability of packet drops due to fading as well as due to buffer over flow are high. Therefore, increasing power of the link will have minimal impact on the probability of packet drops. Hence, we use ECE flag 11 and drops the $cwnd$ size by half.

To determine both short-term and long-term congestion, we use the average queue q_{avg} and the transmission power P_l. We define q_{lim} as the limiting queue size at the wireless router and divide the range of power transmission $P_{l_{Min}}$ - $P_{l_{Max}}$ into two levels, $Region\ A$ and $Region\ B$. The parameter q_{lim} and the division point for $Region\ A$ and $Region\ B$ can be determined experimentally. Based on the values of q_{lim} and the transmission power of the nodes, we define the ECE flags and the actions to be taken in the modified ECN scheme, which we present in Table 7.4.

Table 7.4: Summary of ECE Flags used in the Modified ECN Framework

q_{avg}	P_l	ECE Flag	Action
$\geq q_{lim}$	$Region\ A$	11	$cwnd(t+1) = cwnd(t)/2$
$\geq q_{lim}$	$Region\ B$	10	$cwnd(t+1) = cwnd(t)$
$< q_{lim}$	$Region\ A$	10	$cwnd(t+1) = cwnd(t)$
$< q_{lim}$	$Region\ B$	01	$cwnd(t+1) = cwnd(t) + 1$

Chapter 8

Conclusions and Future Work

The ever-widening customer base and growing demand for higher data rates in wireless networks have motivated researchers to design better resource allocation schemes. However, the time varying nature of wireless channel has posed key challenges while designing such schemes. In this thesis, we have proposed various cross-layer resource allocation schemes. These schemes have exploited the knowledge of the time varying nature of wireless channel, in order to provide high throughput. We have considered applications of both real-time services as well as best effort services in this thesis. We have broadly categorized this thesis into two parts: (I) *Channel and Transport Layer aware Scheduling in Infrastructure-based IEEE 802.16 Networks* and (II) *Channel aware Congestion Control in Infrastructure-less Ad-hoc Networks.*

The first part of the thesis concentrates on uplink scheduling techniques for an infrastructure-based multipoint-to-point wireless networks like IEEE 802.16 network. In Chapter 2, we have investigated the performance of TCP-based and real-time applications in IEEE 802.16-2004 deployed networks of a leading telecom operator in India. We have conducted various experiments both in the laboratory test-bed setup as well as in the live-network setup. In the laboratory test-bed, we have performed experiments with and without ARQ in the system. From these experiments, we have observed that the throughput achieved by the TCP-based applications with ARQ is higher as compared to that achieved without ARQ for similar channel states. We have noted that the percentage of re-transmission of TCP packet drops drastically when ARQ is employed in the system. As a result there is an increase in throughput. We have also compared the performance of TCP-based applications with that of UDP-based applications. From these experiments, we have observed that the throughput achieved by UDP-based application

is substantially higher (more than even 100% in some experiments) than that of TCP-based application for similar channel states, irrespective of the network types. This is because, the Weight-based scheduler used for uplink and downlink scheduling provides equal number of slots to both TCP and UDP applications for similar channel states. The weights of both these applications are the same. Higher throughput of UDP-based applications in comparison with TCP-based applications may be attributed to the slot utilization of both these applications. We have observed that the throughput of TCP-based application suffers in the presence of simultaneous UDP-based applications. In addition, we have noticed that the Weight-based scheduling scheme implemented in the IEEE 802.16 deployed network does not guarantee any scheduling delay to the applications. Since the packets of real-time applications are associated with deadline of scheduling, there is also a need to design scheduling schemes for such applications, in order to provide delay guarantee. This has motivated us to investigate scheduling schemes which are specific to real-time and TCP-based applications.

In Chapter 2, we have also provided an overview of fading in wireless channel and brief description of IEEE 802.16 standard. We have reviewed literature in scheduling in IEEE 802.16 networks and have illustrated various scheduling schemes and fairness issues related to wireless networks.

In Chapter 3, we have proposed a polling based scheduling scheme for real-time applications of *rtPS* service class in IEEE 802.16. In this scheme, we have formulated an optimal method to determine the polling interval, such that the packet drops due to delay violation is minimized and fairness is maintained. The O-DRR scheduling scheme that we have proposed, considers the deadline associated with the Head of the Line (HoL) packets of the participating flows. However, by assigning slots based only on deadline may result in unfairness. Therefore, we have considered the concept of deficit counters of DRR scheduler while scheduling. To evaluate the performance of O-DRR scheduler, we have conducted exhaustive simulations with different loads, traffic types, polling intervals and log-normal shadowing. We have considered both fixed packet sized Video traffic and variable packet sized Pareto traffic (web traffic) in our simulations. From the simulation results, we have observed a concave relationship between the percentage of packets dropped and the polling epoch k. The high percentage of drops at small k is due to the large overheads of polling, whereas it is due to deadline expiry at high k. To demonstrate the robustness of the O-DRR scheduler, we have performed simulations at different

values of log-normal shadowing. From the results, we have noted that the Jain's Fairness Index (JFI) obtained for both single as well as multiclass traffic (Video, Pareto and Mixed) is above 98% even when the load of the system is above 95% and the variance of log-normal shadowing (σ) varying between 4 to 12 dB.

We have also compared the performance of O-DRR scheduler with that of Round Robin (RR) scheduler. The O-DRR scheduler outperforms RR scheduler in both achieved throughput and fairness. We have demonstrated that at 95% load, the percentage of improvement in packet drops due to deadline violation of O-DRR scheduler over RR scheduler is 37%, 27% and 18% for Video, Mixed and Pareto traffic. Moreover, the JFI achieved by the O-DRR scheduler is more than that of RR scheduler at all load conditions. The polling epoch obtained in our scheme has ensured bandwidth assignment in most of the cases.

Moving into the next stage of our analysis, in Chapter 4, we have proposed fixed modulation based scheduling schemes for TCP-based applications that belong to *nrtPS* and *BE* services of IEEE 802.16. We have proposed two scheduling schemes - TWUS and DTWUS. TWUS considers $cwnd$ of the TCP flows along with RTT, whereas DTWUS considers $cwnd$ along with RTT and TCP timeout of the TCP flows. These schemes are polling based schemes. In this chapter, we have highlighted the need for polling based scheduling rather than contention based scheduling. We have argued that the polling epoch for both TWUS and DTWUS scheduler should be of the order of one RTT. This is to ensure minimum packet drops due to TCP timeouts. For implementation, we have employed the request-grant mechanism of IEEE 802.16 standard. In this scheme, each user determines its requirement based on its current $cwnd$ size and communicates it to the BS at the time of polling. To avoid unfairness due to scheduling based only on requirements, we have employed deficit counters similar to that of DRR scheme.

To evaluate the performance of the TCP-aware schedulers, we have conducted extensive simulations. We have also compared the performance of TCP-aware scheduler with that of popular RR scheduler and proprietary Weight-based scheduler. For comparison, we have implemented RR, channel dependent Weight-based (WB (CD)) and channel independent Weight-based (WB (CI)) schedulers at the BS. From the simulation results, we have observed that both average $cwnd$ size and average TCP throughput obtained by the TCP-aware schedulers are higher than that of Weight-based schedulers (WB (CD) and WB (CI)). We have demonstrated that the gain in average $cwnd$ size over WB (CD) varies in the range of 21-34%, whereas it is

in the range of 37-88% over WB (CI) scheduler and 21-27% over RR scheduler. We have also compared the slot utilization of TCP aware schedulers, RR scheduler and Weight-based schedulers. The higher throughput and $cwnd$ size in TCP-aware schedulers is attributed to the higher slot utilization of TCP-aware schemes over others. Moreover, the under-utilization of slots in RR and Weight-based schedulers are substantial, whereas it is zero in TCP-aware schedulers. From the simulations, we have observed that the TCP-aware schedulers outperform RR and WB schedulers in JFI at all fading conditions.

Further, in Chapter 5, we have proposed adaptive modulation based TCP-aware scheduling schemes. Since AMC can be used to achieve high spectral efficiency in fading channels, we exploit AMC for TCP-aware scheduling. With AMC, the BS adapts to a suitable modulation scheme such that the overall system capacity can be increased. In this chapter, we have described a method to implement AMC based TCP-aware scheduling schemes in an IEEE 802.16 network. Similar to Chapter 4, in this chapter too, we have implemented adaptive modulation based RR and Weight-based scheduling schemes at the BS. From the simulation results, we have demonstrated that the proposed TCP-aware scheduling schemes perform better than RR and WB schedulers in terms of slot utilization, fairness and throughput. By varying the standard deviation (σ) of Log-normal shadowing, we have also verified the robustness of the TCP-aware schedulers. Further, we have noted that the proposed schemes succeed in stabilizing the $cwnd$ size. With adaptive modulation, higher rate of transmission is achieved as compared to fixed modulation. However, this higher transmission rate of adaptive modulation is not directly reflected on average TCP throughput and $cwnd$ size. This is due to the fact that the time scale of change of $cwnd$ size is slower than that of the rate of transmission. Therefore, even though the rate of transmission is improved by 80-90%, the average $cwnd$ size is improved by 50-55% only. In this chapter, we have also analytically derived the average TCP $send\,rate$ and have validated the correctness of our analytical results with simulations. Further, we have discussed the properties of TCP-aware scheduler and have proved that the wireless fairness measure obtained in TCP-aware scheduler is bound by a small constant.

The second part of the thesis deals with channel aware congestion control in infrastructureless wireless ad-hoc networks. In this scheme, our prime concern is to provide higher TCP throughput and optimum transmission power in an ad-hoc network. In Chapter 6, we have reviewed the representative literature on congestion control in wired and wireless networks.

We have also discussed some open problems of cross-layer based congestion control in ad-hoc networks, such as life time of the network and energy constraints of the wireless nodes.

We have formulated a joint congestion and power control problem for a CDMA ad-hoc network in Chapter 7. We have discussed congestion cost, energy cost and have attempted to solve the optimization problem in an iterative way. In this scheme, each participating node determines its optimal transmission power in an iterative manner and supports the congestion control scheme implemented in the network to achieve higher throughput. By controlling transmission power along with congestion control, we have demonstrated that congestion in the network can be minimized effectively. We have performed various experiments to determine the efficiency of our framework. The proposed algorithm converges fast for small values of step sizes and for addition and/or deletion of flows into the network. As expected, the channel aware congestion control technique provided stabilized throughput and low transmission power for reasonably good channel states. We have noted that the gain in average $cwnd$ size achieved by the proposed scheme over the conventional fixed power based congestion control scheme is around 27%. However, if the channel gains are poor, then there would be more losses due to poor channel resulting in a significant increase in the network cost. We have also observed that as the channel gain deteriorates, transmission power decreases (similar to Opportunistic scheduling). In such cases, the improvement of the cross-layer scheme over the traditional congestion control scheme is not significant. We have also observed that there is a significant improvement in transmission power of the proposed scheme; 44% when both energy and congestion cost are considered and 7% when only congestion cost is considered. To implement the cross-layer congestion control scheme, we have proposed an Explicit Congestion Notification (ECN) based technique in this chapter. This is a modified ECN approach involving both ECN-Capable Transport (ECT) code points 0 and 1. It uses 2-bit ECE flag as a function of both average queue size and transmission power instead of the usual 1-bit ECE flag.

To summarize, we have investigated cross-layer based resource allocation schemes for real-time as well as best effort services in wireless networks. Particularly, the first part of the thesis deals with channel and Transport layer aware scheduling schemes for the uplink of an infrastructure-based multipoint-to-point IEEE 802.16 network. In the second part of the thesis, we have concentrated on channel aware congestion control schemes for infrastructure-less ad-hoc networks. An effective implementation of these schemes can lead to higher through-

put, maximization of resource utilization and cost effective solutions. Various areas for further investigation have emerged from this thesis. In the next section, we discuss some of these possibilities as future work of this thesis.

8.1 Future Work

For the scheduling schemes proposed in Chapter 3, there emerge several possibilities for further research. One extension would be to look at adaptive modulation and coding schemes between a SS and BS. Another scope for study is the effect of location-dependent channel variations on the performance of the proposed scheme. The effect of ARQ and Hybrid-ARQ (HARQ) also needs to be analyzed along with the scheduling.

For the TCP-aware scheduling schemes, we have assumed that the downlink does not have any bandwidth constraint. In practice this assumption may not hold true. Hence, the effect of downlink congestion and ACK drops on the uplink scheduling and TCP throughput analysis need to be considered while designing a scheduler. Similarly, the effect of ARQ and Hybrid-ARQ (HARQ) should also be investigated along with scheduling. By the inclusion of ARQ and HARQ to our model, even though the scheduling delay increases, increase in TCP throughput is expected as the packet loss probability decreases. This requires further investigation in the TCP *send rate* or throughput determination. We have also not considered the effect of mobility into the performance of TCP-aware scheduling.

While considering the optimization framework in Chapter 7, we have assumed that the routes of the network are static. However, in practice routes can be altered dynamically. Therefore, the optimization framework requires to be extended with dynamic routing. In addition, the channel aware congestion control scheme can be extended to practical IEEE 802.11 based ad-hoc networks. This requires modelling and incorporation of MAC layer activities into the optimization framework.

Bibliography

[1] LAN/MAN Standards Committee, *IEEE Standards for Local and Metropolitan Area Network:Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*. IEEE Computer Society, 1999.

[2] W. Stallings, *Wireless Communications and Networks*. Prentice Hall, 2nd ed., 2005.

[3] LAN/MAN Standards Committee, *IEEE Standards for Local and Metropolitan Area Network: Part 16: Air Interface for Fixed Broadband Wireless Access Systems*. IEEE Computer Society and IEEE Microwave Theory and Techniques Society, May 2004.

[4] LAN/MAN Standards Committee, *IEEE Standards for Local and Metropolitan Area Network: Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems (Amendment2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands)*. IEEE Computer Society and IEEE Microwave Theory and Techniques Society, February 2006.

[5] J. G. Andrews, A. Ghosh, and R. Muhamed, *Fundamentals of WiMAX: Understanding Broadband Wireless Networking*. Prentice Hall, 2007.

[6] "Third Generation Partnership Project." [Online] Available: `http://www.3gpp.org/specs/specs.htm`.

[7] "Third Generation Partnership Project 2." [Online] Available: `http://www.3gpp2.org/Public_html/specs/index.cfm`.

[8] L. Alonso and R. Agusti, "Automatic Rate Adaptation and Energy-Saving Mechanisms Based on Cross-Layer Information for Packet-Switched Data Networks," *IEEE Radio Communications*, pp. S15–S20, March 2004.

[9] M. van der Schaar and N. Sai Shankar, "Cross-Layer Wireless Multimedia Transmission: Challenges, Principles, and New Paradigms," *IEEE Wireless Communications*, vol. 2, pp. 50–58, August 2005.

[10] X. Zhang, J. Tang, H.-H. Chen, S. Ci, and M. Guizani, "Cross-Layer-Based Modeling for Quality of Service Guarantees in Mobile Wireless Networks," *IEEE Communications Magazine*, vol. 44, pp. 100–106, January 2006.

[11] X. Qiuyan and M. Hamdi, "Cross Layer Design for IEEE 802.11 WLANs: Joint Rate Control and Packet Scheduling," in *Proc. of IEEE LCN*, pp. 624–631, November 2005.

[12] Y. Fang and A. B. McDonald, "Cross-Layer Performance Effects of Path Coupling in Wireless Ad Hoc Networks: Power and Throughput Implications of IEEE 802.11 MAC," in *Proc. of IEEE PCCC*, pp. 281–290, 2002.

[13] X. Liu, E. K. P. Chong, and N. B. Shroff, "A Framework for Opportunistic Scheduling in Wireless Networks," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 41, no. 4, pp. 451–474, 2003.

[14] X. Qin and R. A. Berry, "Opportunistic Splitting Algorithms for Wireless Networks," in *Proc. of IEEE INFOCOM*, vol. 3, pp. 1662–1672, March 2004.

[15] R. Knopp and P. A. Humblet, "Information Capacity and Power Control in Single-Cell Multiuser Communications," in *Proc. of IEEE ICC*, vol. 1, pp. 331–335, June 1995.

[16] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks," *IEEE Transactions on Automatic Control*, vol. 37, pp. 1936–1938, December 1992.

[17] S. S. Kulkarni and C. Rosenberg, "Opportunistic Scheduling: Generalizations to Include Multiple Constraints, Multiple Interfaces, and Short Term Fairness," *Wireless Networks*, vol. 11, no. 5, pp. 557–569, 2005.

[18] M. J. Neely, "Order Optimal Delay for Opportunistic Scheduling in Multi-User Wireless Uplinks and Downlinks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 5, pp. 1188–1199, 2008.

[19] A. K. F. Khattab and K. M. F. Elsayed, "Opportunistic Scheduling of Delay Sensitive Traffic in OFDMA-Based Wireless," in *Proc. of IEEE WoWMoM*, pp. 279–288, June 2006.

[20] Y. Liu and E. Knightly, "Opportunistic Fair Scheduling over Multiple Wireless Channels," in *Proc. of IEEE INFOCOM*, vol. 2, pp. 1106–1115, March 2003.

[21] Y. Yu and G. B. Giannakis, "Opportunistic Medium Access for Wireless Networking Adapted to Decentralized CSI," *IEEE Transactions on Wireless Communications*, vol. 5, no. 6, pp. 1445–1455, 2006.

[22] M. C. Chan and R. Ramjee, "Improving TCP/IP Performance over Third Generation Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 4, pp. 430–443, 2008.

[23] R. Ludwig and R. H. Katz, "The Eifel algorithm: Making TCP robust Against Spurious Retransmissions," *ACM SIGCOMM Computer Communication Review*, vol. 30, pp. 30–36, August-September 2000.

[24] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, and A. Viterbi, "CDMA/HDR: A Bandwidth Efficient High Speed Wireless Data Service for Nomadic Users," *IEEE Communications Magazine*, vol. 38, no. 7, pp. 70–77, 2000.

[25] B.-J. Kwak, N.-O. Song, and L. E. Miller, "On the Scalability of Ad-hoc Networks," *IEEE Communications Letters*, vol. 8, pp. 503–505, August 2004.

[26] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," *ACM SIGCOMM Computer Communication Review*, vol. 19, pp. 1–12, September 1989.

[27] S. Golestani, "A Self-Clocked Fair Queuing Scheme for Broadband Applications," in *Proc. of IEEE INFOCOM*, pp. 636–646, June 1994.

[28] J. C. R. Bennett and H. Zhang, "WF^2Q: Worst-case Fair Weighted Fair Queueing," in *Proc. of IEEE INFOCOM*, pp. 120–128, March 1996.

[29] M. Shreedhar and G. Varghese, "Efficient Fair Queueing using Deficit Round Robin," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 375–385, June 1996.

[30] M. Chiang and R. Man, "Jointly Optimal Congestion Control and Power Control in Wireless Multi-hop Networks," in *Proc. of IEEE GLOBECOM*, December 2003.

[31] M. Chiang, "Balancing Transport and Physical Layers in Wireless Multi-hop Networks: Jointly Optimal Congestional Control and Power Control," *IEEE Journal on Selected Areas in Communication*, vol. 23, pp. 104–116, January 2005.

[32] M. Chiang, "To Layer or Not To Layer: Balancing Transport and Physical Layers in Wireless Multihop Networks," in *Proc. of IEEE INFOCOM*, March 2004.

[33] S. H. Low and D. E. Lapsley, "Optimization Flow Control-I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, vol. 7, November 1999.

[34] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," *RFC-3168*, September 2001.

[35] H. S. Wang and N. Moayeri, "Modeling, Capacity, and Joint Source/Channel Coding for Rayleigh Fading Channels," in *Proc. of IEEE VTC*, pp. 473–479, May 1993.

[36] G. S. Prabhu and P. M. Shankar, "Simulation on Flat Fading Using MATLAB for Classroom Instruction," *IEEE Transactions on Education*, vol. 45, pp. 19–25, February 2002.

[37] B. Sklar, "Rayleigh Fading Channels in Mobile Digital Communication Systems. I. Characterization," *IEEE Communications Magazine*, vol. 35, no. 7, pp. 90–100, 1997.

[38] B. Sklar, "Rayleigh Fading Channels in Mobile Digital Communication Systems. II. Mitigation," *IEEE Communications Magazine*, vol. 35, no. 7, pp. 102–109, 1997.

[39] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2006.

[40] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[41] "Understanding Wi-Fi and WiMAX as Metro-Access Solutions," *Intel Corporation, White Paper*, August 2004. [Online] Available: `http://whitepapers.silicon.com`.

[42] S. Keshav, *An Engineering Approach to Computer Networking*. Addison-Wesley Professional Computing Series, 1997.

[43] L. L. Peterson and B. S. Davie, *Computer Networks: A Systems Approach*. Harcourt (India) Private Limited, 2000.

[44] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Single-Node Case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, 1993.

[45] S. Lu, V. Bharghavan, and R. Srikant, "Fair Scheduling in Wireless Packet Networks," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 473–489, August 1999.

[46] N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed Fair Scheduling in a Wireless LAN," in *Proc. of ACM MobiCom*, pp. 167–178, August 2000.

[47] P. Bhagwat, P. Bhattacharya, A. Krishna, and S. K. Tripathi, "Enhancing throughput over wireless LANs using Channel State Dependent Packet Scheduling," in *Proc. of IEEE INFOCOM*, pp. 1133–1140, March 1996.

[48] W. K. Wong, H. Tang, and V. C. M. Leung, "Token Bank Fair Queuing: a New Scheduling Algorithm for Wireless Multimedia Services," *International Journal of Communication Systems*, vol. 08, pp. 591–614, September 2004.

[49] C. Cicconetti, A. Erta, L. Lenzini, and E. Mingozzi, "Performance Evaluation of the IEEE 802.16 MAC for QoS Support," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 26–38, January 2007.

[50] C. Cicconetti, L. Lenzini, E. Mingozzi, and C. Eklund, "Quality of Service Support in IEEE 802.16 Networks," *IEEE Network*, vol. 20, pp. 50–55, March-April 2006.

[51] K. Wongthavarawat and A. Ganz, "Packet Scheduling for QoS Support in IEEE 802.16 Broadband Wireless Access Systems," *International Journal of Communication Systems*, vol. 16, pp. 81–96, February 2003.

[52] N. Liu, X. Li, C. Pei, and B. Yang, "Delay Character of a Novel Architecture for IEEE 802.16 Systems," in *Proc. of IEEE PDCAT*, pp. 293–296, 2005.

[53] M. Hawa and D. W. Petr, "Quality of Service Scheduling in Cable and Broadband Wireless Access Systems," in *Proc. of IEEE IWQoS*, pp. 247–255, 2002.

[54] D. H. Kim and C. G. Kang, "Delay Threshold-based Priority Queueing Scheme for Packet Scheduling in Mobile Broadband Wireless Access System," in *Proc. of IEEE HPCC*, pp. 305–314, 2005.

[55] J. M. Ku, S. K. Kim, S. H. Kim, S. Shin, J. H. Kim, and C. G. Kang, "Adaptive Delay Threshold-based Priority Queueing Scheme for Packet Scheduling in Mobile Broadband Wireless Access System," in *Proc. of IEEE WCNC*, pp. 1142–1147, 2006.

[56] J. Chen, W. Jiao, and H. Wang, "A Service Flow Management Strategy for IEEE 802.16 Broadband Wireless Access Systems in TDD Mode," in *Proc. of IEEE ICC*, vol. 5, pp. 3422–3426, June 2005.

[57] J. Chen, W. Jiao, and H. Wang, "An Integrated QoS Control Architecture for IEEE 802.16 Broadband Wireless Access Systems," in *Proc. of IEEE GLOBECOM*, vol. 6, pp. 3330–3335, December 2005.

[58] W. K. Wong, H. Tang, S. Guo, and V. C. M. Leung, "Scheduling Algorithm in a Point-to-Multipoint Broadband Wireless Access Network," in *Proc. of IEEE VTC*, vol. 3, pp. 1593–1597, October 2003.

[59] D. Niyato and E. Hossain, "Queue-Aware Uplink Bandwidth Allocation and Rate Control for Polling Service in IEEE 802.16 Broadband Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 668–679, June 2006.

[60] F. Hou, P.-H. Ho, X. S. Shen, and A.-Y. Chen, "A Novel QoS Scheduling Scheme in IEEE 802.16 Networks," in *Proc. of IEEE WCNC*, pp. 2457–2462, 2007.

[61] A. Sayenko, O. Alanen, and T. Hämäläinen, "Scheduling solution for the IEEE 802.16 base station," *Computer Networks*, vol. 52, no. 1, pp. 96–115, 2008.

[62] S. Kim and I. Yeom, "TCP-Aware Uplink Scheduling for IEEE 802.16," *IEEE Communications Letters*, vol. 11, pp. 146–148, February 2007.

[63] L. Lenzini, E. Mingozzia, and G. Stea, "Tradeoffs Between Low Complexity, Low Latency and Fairness With Deficit Round-Robin Schedulers," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 681–693, August 2004.

[64] M. Dianati, X. S. Shen, and S. Naik, "A New Fairness Index for Radio Resource Allocation in Wireless Networks," in *Proc. of IEEE WCNC*, pp. 712–717, March 2005.

[65] R. Jain, D.-M. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination For Resource Allocation in Shared Computer Systems," *Technical Report TR-301, DEC Research Report*, September 1984.

[66] K. Norlund, T. Ottosson, and A. Brunstorm, "TCP fairness measures for scheduling algorithms in wireless networks," in *Proc. of IEEE QShine*, p. 20, August 2005.

[67] E. Biglieri, J. Proakis, and S. Shamai, "Fading Channels: Information-Theoretic and Communications Aspects," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 2619–292, October 1998.

[68] "Matlab version-6." [Online] Available: `http://www.mathworks.com`.

[69] P. Droz and J.-Y. L. Boudec, "A High-Speed Self-Similar ATM VBR Traffic Generator," in *Proc. of IEEE GLOBECOM*, pp. 586–590, November 1996.

[70] B. Baynat, S. Doirieux, G. Nogueira, M. Maqbool, and M. Coupechoux, "An Efficient Analytical Model for WiMAX Networks with Multiple Traffic Profiles," in *Proc. of Mobility Conference*, pp. 1–6, ACM, 2008.

[71] H.-O. Peithen, H. Jurgens, and D. Saupe, *Chaos and Fractals, New Frontiers of Science*. Springer Verlag, 1992.

[72] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," *RFC-2581*, April 1999.

[73] R. Braden, V. Jacobson, and L. Zhang, "TCP Extensions for Highspeed Paths," *RFC-1185*, October 1990.

[74] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options," *RFC-2018*, April 1996.

[75] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modelling TCP Reno Performance: A Simple Model and its Empirical Validation," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 133–145, April 2000.

[76] T. M. Heikkinen, "On Congestion Pricing in a Wireless Network," *Wireless Networks*, vol. 8, no. 4, pp. 347–354, 2002.

[77] R. Srikant, *The Mathematics of Internet Congestion Control*. Birkhauser, Boston, 1st ed., 2003.

[78] C. Courcoubetis and R. Weber, *Pricing Communication Networks: Economics, Technology and Modeling*. John Wiley and Sons, 2003.

[79] S. Ryu, C. Rump, and C. Qiao, "Advances in Internet Congestion Control," *IEEE Communications Surveys and Tutorials*, vol. 3, pp. 28–39, 2003.

[80] V. Jacobson, "Congestion Avoidance and Control," *ACM SIGCOMM Computer Communication Review*, pp. 314–329, August 1988.

[81] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," April 1999. RFC-2582.

[82] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," *ACM SIGCOMM Computer Communication Review*, vol. 24, pp. 24–35, August - September 1994.

[83] U. Hengartner, J. Bolliger, and T. Gross, "TCP Vegas Revisited," in *Proc. of IEEE INFOCOM*, pp. 1546–1555, March 2000.

[84] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation Based Congestion Control for Unicast Applications," *ACM SIGCOMM Computer Communication Review*, vol. 30, pp. 43–56, March 2000.

[85] T. V. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 336–350, June 1997.

[86] S. J. Golestani and S. Bhattacharyya, "A Class of End-to-End Congestion Control Algorithms for the Internet," in *Proc. of ICNP*, pp. 137–151, IEEE Computer Society, October 1998.

[87] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, 1993.

[88] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *Proc. of IEEE INFOCOM*, vol. 3, pp. 1346–1355, March 1999.

[89] S. Kunniyur and R. Srikant, "A Time-Scale Decomposition Approach to Adaptive Explicit Congestion Notification (ECN) Marking," *IEEE Transactions on Automatic Control*, vol. 47, June 2002.

[90] S. Kunniyur and R. Srikant, "End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks," *IEEE/ACM Transactions on Networking*, vol. 11, October 2003.

[91] F. P. Kelly, "Charging and Rate Control for Elastic Traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, January 1997. [Online] Corrected version of the original paper is available at: `http://www.statslab.cam.ac.uk/~frank/elastic.ps`.

[92] R. J. Gibbens and F. P. Kelly, "Resource Pricing and Evolution of Congestion Control," *Automatica*, vol. 35, December 1999.

[93] F. P. Kelly, "Fairness and Stability of End-to-End Congestion Control," *European Journal of Control*, vol. 9, pp. 159–176, 2003.

[94] S. Shenker, "Fundamental Design Issues for the Future Internet," *IEEE Journal on Selected Areas in Communication*, vol. 13, September 1995.

[95] S. Shakkottai, T. S. Rappaport, and P. C. Karlsson, "Cross-Layer Design for Wireless Networks," *IEEE Communications Magazine*, pp. 74–80, October 2003.

[96] Y. Tian, K. Xu, and N. Ansari, "TCP in Wireless Environments: Problems and Solutions," *IEEE Radio Communications*, pp. S27–S32, March 2005.

[97] V. Raghunathan and P. R. Kumar, "A counterexample in congestion control of wireless networks," *Performance Evaluation*, vol. 64, no. 5, pp. 399–418, 2007.

[98] V. Srivastava and M. Motani, "Cross-Layer Design: A Survey and the Road Ahead," *IEEE Communications Magazine*, vol. 43, no. 12, pp. 112–119, 2005.

[99] S. Kopparty, S. V. Krishnamurthy, M. Faloutsos, and S. K. Tripathi, "Split TCP for Mobile Ad Hoc Networks," in *Proc. of IEEE GLOBECOM*, pp. 138–142, November 2002.

[100] J. L. Sun and S. Singh, "ATCP: TCP for Mobile Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 1300–1315, 1999.

[101] D. Kim, C.-K. Toh, and Y. Choi, "TCP-BuS: Improving TCP Performance in Wireless Ad Hoc Networks," in *Proc. of IEEE ICC*, pp. 1707–1713, June 2000.

[102] E. Amir, H. Balakrishnan, S. Seshan, and R. H. Katz, "Efficient TCP over Networks with Wireless Links," in *Proc. of HotOS*, p. 35, 1995.

[103] Umut Akyol and Matthew Andrews and Piyush Gupta and John Hobby and Iraj Saniee and Alexander Stolyar, "Joint Scheduling and Congestion Control in Mobile Ad-Hoc Networks," in *Proc. of IEEE INFOCOM*, pp. 619–627, April 2008.

[104] Lijun Chen and Steven H. Low and Mung Chiang and John C. Doyle, "Cross-Layer Congestion Control, Routing and Scheduling Design in Ad Hoc Wireless Networks," in *Proc. of IEEE INFOCOM*, pp. 1–13, April 2006.

[105] M. Chiang, "Distributed Network Control through Sum Product Algorithm on Graphs," in *Proc. of IEEE GLOBECOM*, pp. 2395–2399, November 2002.

[106] S. H. Low, "A Duality Model of TCP and Queue Management Algorithms," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 526–536, August 2003.

[107] S. H. Low, F. Paganini, and J. C. Doyle, "Internet Congestion Control," *IEEE Control Systems Magazine*, pp. 28–43, February 2002.

[108] F. Paganini, Z. Wang, J. C. Doyle, and S. H. Low, "Congestion Control for High Performance, Stability and Fairness in General Networks," *IEEE/ACM Transactions on Networking*, vol. 13, pp. 43–56, February 2005.

[109] J. Monks, V. Bharghavan, and W. Hwu, "A Power Controlled Multiple Access Protocol for Wireless Packet Networks," in *Proc. of IEEE INFOCOM*, pp. 219–228, April 2001.

[110] A. Muqattash and M. Krunz, "CDMA-based MAC Protocol for Wireless Ad hoc Networks," in *Proc. of ACM MobiHoc*, pp. 153–162, June 2003.

[111] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 2nd ed., 1999.

[112] S. K. Bose, *An Introduction to Queuing Systems*. Kluwer Academic / Plenum Publishers, 2002.

[113] A. Muqattash and M. Krunz, "A Single-Channel Solution for Transmission Power Control in Wireless Ad-hoc Networks," in *Proc. of ACM MobiHoc*, pp. 210–222, May 2004.

List of Publications

1. H. K. Rath, A. Karandikar and V. Sharma, “Adaptive Modulation-based TCP-Aware Uplink Scheduling in IEEE 802.16 Networks”, in *Proc. of IEEE ICC*, May 2008.

2. H. K. Rath, A. Karandikar, “On Cross Layer Congestion Control for CDMA- based Adhoc Networks”, in *Proc. of National Conference on Communications (NCC)*, February 2008.

3. H. K. Rath and A. Karandikar, “On TCP-Aware Uplink Scheduling in IEEE 802.16 Networks”, in *Proc. of 3rd IEEE COMSWARE*, January, 2008.

4. H. K. Rath, A. Bhorkar, V. Sharma, “An Opportunistic Uplink Scheduling Scheme to Achieve Bandwidth Fairness and Delay for Multiclass Traffic in IEEE 802.16 Broadband Wireless Networks”, in *Proc. IEEE GLOBECOM*, November 2006.

5. H. K. Rath, A. Bhorkar, V. Sharma, “An Opportunistic DRR (O-DRR) Uplink Scheduling Scheme for IEEE 802.16-based Broadband Wireless Networks”, in *Proc. of IETE International Conference on Next Generation Networks*, Feb 2006.

6. H. K. Rath, A. Sahoo, A. Karandikar, “Cross Layer Congestion Control Algorithm in Wireless Networks for TCP Reno-2”, in *Proc. of National Conference on Communications (NCC)*, January 2006.